

Using evolutionary algorithms to  
resolve 3-dimensional geometries  
encoded in indeterminate  
data-sets



Graham Rollings

College of Engineering, Mathematics and Physical Sciences

University of Exeter

A thesis submitted to the University of Exeter for the degree of  
*Doctor of Philosophy in Computer Science*

November 2011

---

---

---

## Declaration

This thesis is available for Library use on the understanding that it is copyright material and that no quotation from the thesis may be published without proper acknowledgement.

I certify that all material in this thesis which is not my own work has been identified and that no material has previously been submitted and approved for the award of a degree by this or any other University.

Print Name: .....

Signature: .....

Date: .....

---

## Papers published as a result of this work

The following papers have been co-written by the Author and principal supervisor, Professor David Corne. The work presented in these papers has been expanded and presented in more detail throughout the thesis.

**Rollings GA. and Corne DW. [103]**

**Relative Distance Identification in "Smart Dust" Networks for Environmental Modelling.** 7th WSEAS International. Conference. on Artificial Intelligence, Knowledge Engineering and Databases (AIKED' 08), University of Cambridge, UK, Feb 2008.

Abstract: Smart dust (wireless sensor) networks have many applications in environmental monitoring. In many such applications, the data gathered by the motes is only useful in combination with knowledge of the motes positions, and this needs to be estimated in cases where they are airborne, or otherwise not statically fixed. GPS based location solutions are prohibitively expensive and also present some operational difficulties, so it is important to find others ways to determine location. This is usually termed *Localisation*, a problem made more complex where the position and velocity of the motes are subject to external forces. We investigate this by exploring the use of optimisation algorithms to find relative distance configurations that minimise a straightforward error function based on the Received Signal Strength Indication (RSSI). In this paper we focus on simulating noise-free conditions; although less realistic, this provides information about whether or not this problem can be satisfactorily solved at all in reasonable time, and, if so, for what sizes of mote network will this be possible. We find that for the range of mote clusters sizes considered as a static field model the results are indeed sufficiently encouraging to indicate the need to continue the research.

**Key-Words:** - Smart dust, evolutionary algorithms, environmental monitoring, location estimation.

**Rollings GA. and Corne DW. [104]**

**Intelligent Operators for Localisation of Dynamic Smart Dust Networks.** 8th International Conference on Hybrid Intelligent Systems (HIS2008), Technical University of Catalonia, Barcelona, Spain, September 2008.

Abstract: Wireless Sensor Networks (WSN) of Smart dust motes are becoming increasingly effective in environmental monitoring applications. In some applications, data gathered via WSN are only useful when combined with individual mote positions and time-stamps; if the motes are not static, it is important to find methods for their 3D location estimation from available RSSI signal data. Usually termed *Localisation*, this problem is made more complex when the positions of the motes are subject to

---

external forces. Here we extend our previous work on solving this problem with evolutionary algorithms; we experiment with *geometric-awareness* operators that constrain the positions a WSN mote can occupy to those that have a better probability of maximally contributing to the chromosome fitness. A hybrid of these specialised operators and standard operators is also tested. We conclude that this research direction offers a viable basis for a heuristically directed evolutionary system capable of suitably accurate 3D localisation, thus increasing the possibilities for viable WSN applications.

**Key-Words:** Smart dust, evolutionary algorithms, environmental monitoring, location estimation, heuristically directed evolution, geometric-awareness.

## Acknowledgements

This thesis would not have been possible without the help and support that I received whilst I was registered as a PhD student at the College of Engineering, Mathematics and Physical Sciences, University of Exeter. I would like to express my heartfelt gratitude and thanks to everyone who has helped and supported me in this work.

Within the academic community of the college I would like to thank my supervisors, Professor David Corne and Professor David Wright. On a more personal level I would like to thank the many students I shared an office with for their amusing antics and general conversation. In particular Maximillian Dupenois, Andrew Clarke, Kent McClymont, David Walker, Michael Barclay, Desmond Choo, Lei Wang, Purav Shah, Lari Lampen, Chenghua Lin, Denh Tran, and many others. And also to the many undergraduates to whom I had the interesting challenge of trying to explain some really simple stuff.

I would like to acknowledge Dr. John Eyre and Mr. Alan Douglas (Retd.) of the UK Meteorological Office for their initial interest by allowing Professor David Corne and myself to give several presentations specifically on the meteorological potential of the research proposal as it is definitely in their area of expertise. Their involvement during the early stages was a big factor in gaining the award of the EPSRC project funding as part of a school DTA studentship.

I would also like to thank Mr. Karl Kitchen, Senior Applied Scientist also of the UK Meteorological Office, for his extensive discussions, expertise in mathematical and particulate plume modelling, and for some very generous support over the course of the recent years.

And, of course, considerable thanks must go to all of my family for the seemingly endless financial support necessary to bring the PhD to a successful conclusion.

---

# Abstract

This thesis concerns the developments of optimisation algorithms to determine the relative co-location, (localisation), of a number of freely-flying '*Smart Dust mote*' sensor platform elements using the a non-deterministic data-set derived from the duplex wireless transmissions between elements. Smart dust motes are miniaturised, micro-processor based, electronic sensor platforms, frequently used for a wide range of remote environmental monitoring applications; including specific climate synoptic observation research and more general meteorology.

For the application proposed in this thesis a cluster of the notional smart dust motes are configured to imitate discrete '*Radio Drop Sonde*' elements of the wireless enabled monitoring system in use by meteorological research organisations worldwide. This cluster is modelled in software in order to establish the relative positions during the '*flight*'; the normal mode of deployment for the Drop Sonde is by ejection from an aeroplane into an upper-air zone of interest, such as a storm cloud.

Therefore the underlying research question is, how to track a number of these independent, duplex wireless linked, free-flying monitoring devices in 3-dimensions and time (to give the monitored data complete spatio-temporal validity). This represents a significant practical challenge, the solution applied in this thesis was to generate 3-dimensional geometries using the only '*real-time*' data available; the Radio Signal Strength Indicator (RSSI) data is generated through the '*normal*' duplex wireless communications between motes. Individual RSSI values can be considered as a '*representation of the distance magnitude*' between wireless devices; when collated into a spatio-temporal data-set it '*encodes*' the relative, co-locational, 3-dimensional geometry of all devices in the cluster. The reconstruction, (or decoding), of the 3-dimensional geometries encoded in the spatio-temporal data-set is a complex problem that is addressed through the application of various algorithms. These include, ***Random Search***, and optimisation algorithms, such as the ***Stochastic Hill-climber***, and various forms of ***Evolutionary Algorithm***.

It was found that the performance of the geometric reconstruction could be improved through identification of salient aspects of the modelled environment, the result was heuristic operators. In general these led to a decrease in the time taken to reach a convergent solution or a reduction in the number of candidate search space solutions that must be considered. The software model written for this thesis has been implemented to generalise the fundamental characteristics of an optimisation algorithm and to incorporate them into a generic software framework; this then provides the common code to all model algorithms used.

---

# Contents

<b>Contents</b>	<b>vii</b>
<b>List of Figures</b>	<b>xiii</b>
<b>List of Tables</b>	<b>xvii</b>
<b>I Introduction, Literature review, and Background</b>	<b>1</b>
<b>1 Introduction</b>	<b>3</b>
1.1 Real world motivation . . . . .	3
1.1.1 The Radio Sonde airborne environmental monitoring system . . .	4
1.1.2 Tracking distant objects . . . . .	4
1.1.3 The ‘Smart Dust’ system . . . . .	5
1.1.4 A Smart Dust based Radio Sonde . . . . .	5
1.1.5 Multiple element Radio Sonde . . . . .	5
1.1.6 The software modelling abstraction . . . . .	6
1.2 Principal contributions . . . . .	7
1.3 Chapter list précis . . . . .	9
1.3.1 Chapter 2 : The literature review . . . . .	9
1.3.2 Chapter 3 : Background . . . . .	9
1.3.3 Chapter 4 : Software model design . . . . .	9
1.3.4 Chapter 5 : Run-time base-line using non-optimised algorithms .	9
1.3.5 Chapter 6 : Establishing optimal algorithm parameter settings .	10
1.3.6 Chapter 7 : Optimisation algorithm simulations . . . . .	10
1.3.7 Chapter 8 : Heuristic and hybrid operators . . . . .	10
1.3.8 Chapter 9 : Conclusions and further work . . . . .	10
1.3.9 Appendices . . . . .	11
<b>2 Literature Review</b>	<b>13</b>
2.1 Environmental monitoring using Smart Dust . . . . .	13
2.2 Localisation and Radio Signal Strength Indication . . . . .	16
2.3 An overview of ant colony and swarm optimisation . . . . .	20
2.4 Optimisation algorithms . . . . .	22

## CONTENTS

---

2.4.1	Evolutionary Algorithms . . . . .	23
2.4.2	Evolution Strategies . . . . .	24
2.5	Chapter summary . . . . .	26
<b>3</b>	<b>Background</b>	<b>27</b>
3.1	Defining localisation . . . . .	28
3.2	Relative distance identification . . . . .	31
3.2.1	Historical methods for the determination of relative position . . . . .	31
3.2.2	An overview of contemporary range detection . . . . .	32
3.3	Radio Signal Strength Indication . . . . .	37
3.3.1	Idealised electro-magnetic wave propagation . . . . .	38
3.3.2	Radio Signal Strength Indication in the real-world environment . . . . .	38
3.3.3	Radio Signal Strength Indication as an input data-set . . . . .	41
3.4	A generic Radio Sonde monitoring system . . . . .	43
3.4.1	Sonde monitoring and wireless communication . . . . .	43
3.4.2	Meteorological Radio Sonde synoptic data types . . . . .	43
3.4.3	The types of Radio Sonde devices . . . . .	44
3.4.4	The Radio Sonde . . . . .	45
3.4.5	The Drop Sonde . . . . .	46
3.4.6	The Drift Sonde . . . . .	47
3.4.7	Some general Radio Sonde system limitations . . . . .	48
3.5	An Overview of evolutionary algorithms . . . . .	49
3.5.1	A simple evolutionary algorithm . . . . .	50
3.6	An overview of the object oriented paradigm . . . . .	52
3.6.1	Problem space representation in a computational environment . . . . .	53
3.6.2	The object-oriented design process . . . . .	54
3.7	Chapter summary . . . . .	54
<b>II</b>	<b>Problem Description</b>	<b>55</b>
<b>4</b>	<b>Software model design</b>	<b>57</b>
4.1	The real-world upper-airspace abstraction . . . . .	58
4.1.1	The Clustered Drop Sonde hardware abstraction . . . . .	59
4.1.2	An overview of a clustered Drop Sonde . . . . .	61
4.2	The software modelling system . . . . .	62
4.2.1	Clustered Drop Sonde run-time overview . . . . .	62
4.3	Virtual base-class frameworks . . . . .	65
4.3.1	System processes in a virtual base-class framework . . . . .	65
4.3.2	A practical virtual base-class framework . . . . .	66
4.4	Implementing a geometric reconstruction model . . . . .	69
4.4.1	The complexity of the search space . . . . .	70
4.5	Representing the problem space in a computer . . . . .	73
4.5.1	The algorithm and run-time control . . . . .	76

4.5.2	The allele: the fundamental dimension definition . . . . .	77
4.5.3	The allele: <b>Drop-Sonde</b> element class structure . . . . .	77
4.5.4	The n-dimensional bounding zone and sonde element classes . . . . .	79
4.5.5	The chromosome and associated population classes . . . . .	79
4.5.6	The operators: the virtual class structure . . . . .	80
4.5.7	The operators: The mutation operators . . . . .	81
4.5.8	The operators: the crossover operators . . . . .	82
4.5.9	The objective fitness: 3-dimensional positional error class . . . . .	84
4.6	The data storage, collation and output sub-system . . . . .	93
4.6.1	The DSS elemental class types . . . . .	94
4.6.2	The DVec storage object . . . . .	95
4.6.3	The DVecV dynamic DVec array storage object . . . . .	97
4.6.4	The DVecV Meta-data and statistics interface . . . . .	98
4.6.5	Range-group classifier meta-data classes . . . . .	98
<b>III 'Standard' Algorithm Simulations</b>		<b>105</b>
<b>5</b>	<b>Software model performance base-line using non-optimised algorithms</b>	<b>107</b>
5.1	The simulated algorithm types . . . . .	108
5.2	Run-time parameters . . . . .	109
5.2.1	Initialisation file section: Simulation specific parameters . . . . .	110
5.2.2	Initialisation file section: Generic system parameters . . . . .	111
5.2.3	Overview of the plotted results data . . . . .	112
5.3	Simulation setup . . . . .	113
5.3.1	Minimising directed fitness convergence . . . . .	113
5.3.2	Overview of the run-time sequence . . . . .	114
5.4	Random Landscape Search . . . . .	116
5.4.1	Random Landscape Search algorithm results . . . . .	117
5.5	Random Progression Search . . . . .	120
5.5.1	Random Progression Search algorithm results . . . . .	121
5.5.2	Range Groups and the Weighted Performance Factor . . . . .	122
5.5.3	Standard deviation . . . . .	123
5.6	Summary of the search algorithm results . . . . .	124
<b>6</b>	<b>Establishing optimal algorithm parameter settings</b>	<b>127</b>
6.1	The default parameter values . . . . .	127
6.2	Mutation distance . . . . .	129
6.2.1	Mutation distance - simulation results . . . . .	130
6.3	Chromosome population size . . . . .	131
6.3.1	Chromosome population size - simulation results . . . . .	131
6.4	Selection simulations . . . . .	133
6.4.1	Tournament selection parameter setting . . . . .	133
6.4.2	Tournament percentage selection - simulation results . . . . .	133

## CONTENTS

---

6.5	Selection mode type . . . . .	135
6.5.1	Selection mode type - simulation results . . . . .	136
6.6	Mutation operator type . . . . .	137
6.6.1	Mutation operator type - simulation results . . . . .	138
6.7	Crossover operator type . . . . .	139
6.7.1	Crossover operator type - simulation results . . . . .	140
6.8	Chromosome mutate/accept ratio . . . . .	142
6.8.1	Chromosome mutate/accept ratio - simulation results . . . . .	142
6.9	Summary . . . . .	144
<b>7</b>	<b>Optimisation algorithm simulations</b>	<b>145</b>
7.1	The simulated algorithm types . . . . .	145
7.2	Run-time parameters . . . . .	146
7.2.1	Initialisation file section: simulation specific parameters . . . . .	147
7.2.2	Initialisation file section: generic system parameters . . . . .	149
7.3	Simulation setup . . . . .	150
7.3.1	Overview of the run-time sequence . . . . .	150
7.4	Stochastic hill-climber . . . . .	151
7.4.1	Stochastic hill-climber algorithm results . . . . .	153
7.5	Standard evolutionary algorithm . . . . .	155
7.5.1	Standard evolutionary algorithm results . . . . .	157
7.5.2	Comparison of simulation results . . . . .	161
7.6	Large data-set evolutionary algorithm results . . . . .	165
7.7	Chapter summary . . . . .	167
<b>IV</b>	<b>Search Space Reduction &amp; Heuristic Simulations</b>	<b>169</b>
<b>8</b>	<b>Heuristic and hybrid operators</b>	<b>171</b>
8.1	Search space characteristics . . . . .	171
8.2	The simulated algorithm types . . . . .	172
8.3	Run-time parameters . . . . .	172
8.4	The gateway constraint operator . . . . .	172
8.4.1	Constraint constraint operator: design . . . . .	173
8.4.2	Constraint constraint operator: Mode of operation . . . . .	175
8.5	Constraint constraint operator: EA simulation results . . . . .	178
8.6	Hybrid operator simulations . . . . .	180
8.6.1	Standard Gateway Constraint Operator hybrid results . . . . .	181
8.6.2	Hybrid operator and standard EA comparative results . . . . .	184
8.7	Chapter summary . . . . .	186

<b>V</b>	<b>Conclusions</b>	<b>189</b>
<b>9</b>	<b>Conclusions and further work</b>	<b>191</b>
9.1	Conclusions . . . . .	191
9.1.1	Global climate change . . . . .	192
9.2	Conclusions on the work of the thesis . . . . .	192
9.2.1	Part I: Introduction, literature review and background . . . . .	192
9.2.2	Part II: The problem description . . . . .	194
9.2.3	Part III: the ‘ <i>standard</i> ’ algorithm simulations . . . . .	196
9.2.4	Part IV: Search space reduction & Heuristic Simulations . . . . .	199
9.3	Further work . . . . .	200
9.3.1	The error model . . . . .	201
9.3.2	Field trials . . . . .	202
9.3.3	Search space reduction heuristics . . . . .	204
9.3.4	Heuristic: Decomposition . . . . .	204
9.3.5	Heuristic: Quaternion operators . . . . .	205
9.3.6	Algorithmic structure . . . . .	206
9.4	Final conclusions . . . . .	207
	<b>Appendices</b>	<b>209</b>
<b>A</b>	<b>Crossbow Technology™ Smart Dust system</b>	<b>211</b>
A.1	The Smart Dust elements . . . . .	212
A.1.1	The Crossbow MICA2DOT Smart Dust mote . . . . .	212
A.1.2	The Crossbow MICA2/MICAz Smart Dust mote . . . . .	213
A.2	The Smart Dust interface devices . . . . .	215
A.2.1	MIB510 RS232 Serial Interface Board . . . . .	216
A.2.2	MIB520 Universal Serial Bus (USB Interface Board . . . . .	217
A.2.3	MIB600 Ethernet Gateway Interface Board . . . . .	217
A.3	Stargate Processing module . . . . .	218
<b>B</b>	<b>Smart Dust as an automatic weather station</b>	<b>223</b>
B.1	A simple automatic monitoring element . . . . .	223
B.1.1	Pressure Sensor and Analogue to Digital Converter example . . . . .	225
B.1.2	A smart dust system sonde implementation . . . . .	225
<b>C</b>	<b>The software model initialisation file</b>	<b>231</b>
C.1	File section: Simulation specific parameters . . . . .	231
C.1.1	Generic simulation specific parameters . . . . .	232
C.1.2	Hybrid operator simulation parameters . . . . .	232
C.1.3	Evolutionary algorithm parameters . . . . .	233
C.1.4	Evolutionary algorithm parameters: fixed origin constraint . . . . .	236
C.2	File section: Generic specific parameters . . . . .	237
C.2.1	Debug parameters . . . . .	237

## CONTENTS

---

C.2.2	Example Initialisation file . . . . .	250
<b>D</b>	<b>The software input data-set file</b>	<b>257</b>
D.1	Header Blocks. . . . .	258
D.1.1	Data Block. . . . .	259
<b>E</b>	<b>Example software model results file.</b>	<b>263</b>
E.1	Header block. . . . .	263
E.2	Data Block. . . . .	263
E.3	Statistics block. . . . .	264
<b>F</b>	<b>RSSI input data-set neutrality test</b>	<b>267</b>
F.1	Objective fitness plots . . . . .	268
F.2	Standard deviation plots . . . . .	269
<b>VI</b>	<b>References</b>	<b>271</b>
	<b>References</b>	<b>273</b>

# List of Figures

3.1	Generic OED definition . . . . .	28
3.2	Definitions that refer to the localisation of physical objects . . . . .	29
3.3	Definitions that refer to the location of a functional property . . . . .	30
3.4	<b>A</b> Definition of localisation derived from robotics . . . . .	30
3.5	Radar based triangulation . . . . .	35
3.6	The propagation inverse-square law . . . . .	38
3.7	Synchronised inter-element packet format description . . . . .	42
3.8	Synchronised inter-element packet format definition . . . . .	42
3.9	A generic Radio Sonde . . . . .	45
3.10	The Vaisala RD94™ Radio Sonde . . . . .	46
3.11	The UCAR Drift Sonde Deployment System . . . . .	47
3.12	Simple definition of an evolutionary algorithm . . . . .	49
3.13	Simple evolutionary algorithm pseudo-code . . . . .	50
4.1	The Virtual Base-class framework block diagram . . . . .	68
4.2	Simple search space complexity calculation . . . . .	71
4.3	A more complex and representative search space complexity calculation . . . . .	71
4.4	Simple search space complexity calculation . . . . .	72
4.5	Simplified overview of the stages in the design process . . . . .	74
4.6	Upper airspace and cluster element overview . . . . .	74
4.7	Upper airspace and cluster element overview . . . . .	75
4.8	Euclidean magnitude between discrete element pair . . . . .	77
4.9	VBF n-dimensional zone (Drop Sonde) representation . . . . .	78
4.10	The Virtual Base-class operator block diagram . . . . .	80
4.11	The derived fitness block diagram . . . . .	84
4.12	Fundamental fitness function types . . . . .	85
4.13	Default objective fitness function . . . . .	86
4.14	Informal objective fitness definition . . . . .	86
4.15	Euclidean magnitude between discrete element pair . . . . .	87
4.16	Single element Cartesian distance magnitude . . . . .	89
4.17	Pseudo-code of objective fitness function . . . . .	91
4.18	Discrete inter-element RSSI magnitude data points . . . . .	92
4.19	Data-set inter-element identifier pairs . . . . .	92

## LIST OF FIGURES

---

4.20	All element objective fitness function . . . . .	93
4.21	Embedded VBF automated data storage class block diagram . . . . .	94
4.22	DVec dynamic array class hierarchical block diagram . . . . .	95
4.23	DVecV dynamic array class hierarchical block diagram . . . . .	98
4.24	Single range-group definition string . . . . .	99
4.25	Discrete range-group function diagram . . . . .	100
4.26	Example RangeGroupV input text string . . . . .	101
4.27	Optional Weighting Factor group definition string . . . . .	102
5.1	Initialisation file: default parameters . . . . .	110
5.2	Initialisation file: Simulation specific parameters . . . . .	111
5.3	Initialisation file: Generic system parameters . . . . .	112
5.4	Run-time pseudo-code overview . . . . .	115
5.5	Pseudo-code of Random Landscape Search algorithm . . . . .	116
5.6	Random Landscape Search Fitness Plot . . . . .	118
5.7	Pseudo-code of Random Progression Search algorithm . . . . .	120
5.8	Random Progression Search Fitness Plot . . . . .	121
5.9	Random Landscape Search standard deviation plot . . . . .	123
5.10	Random Progression Search standard deviation plot . . . . .	123
6.1	Mutation Comparison Plot . . . . .	130
6.2	Population Size Performance Comparison Plot . . . . .	132
6.3	Tournament percentage selection plot . . . . .	134
6.4	Selection mode types . . . . .	136
6.5	Selection mode type comparison plot . . . . .	136
6.6	Mutation operator types . . . . .	138
6.7	Mutation operator comparison plot . . . . .	138
6.8	Single/uniform crossover difference plot (0 to 999 iterations) . . . . .	140
6.9	Single/uniform crossover difference plot (1000 to 10000 iterations) . . . . .	141
6.10	Chromosome mutate/accept ratio comparison plot . . . . .	143
6.11	Simulation parameters . . . . .	144
7.1	Initialisation file: Simulation specific parameters . . . . .	148
7.2	Initialisation file: Disruption event descriptions . . . . .	149
7.3	Initialisation file: Generic system parameters . . . . .	150
7.4	Run-time standard EA & stochastic hill-climber pseudo-code overview . . . . .	151
7.5	Pseudo-code of basic stochastic hill-climber algorithm . . . . .	153
7.6	Stochastic hill-climber fitness plot . . . . .	154
7.7	Pseudo-code of standard EA algorithm . . . . .	156
7.8	Standard EA fitness plot . . . . .	157
7.9	Standard EA fitness plot - abridged X axis . . . . .	157
7.10	Standard EA fitness plot - abridged Y axis . . . . .	158
7.11	Difference plot magnitude zones . . . . .	161
7.12	Random progression search (base-line)/Hill-climber difference fitness plot . . . . .	163

## LIST OF FIGURES

---

7.13	Hill-climber (base-line)/standard EA difference fitness Plot . . . . .	165
7.14	Standard EA Large Dataset Fitness Plot . . . . .	166
8.1	Gateway constraint multiple element depiction . . . . .	174
8.2	Gateway constraint regions . . . . .	175
8.3	Pseudo-code of the radial constraint operator . . . . .	176
8.4	Gateway constrained EA fitness plot . . . . .	178
8.5	Standard EA (base-line)/gateway constrained fitness difference plot . . .	180
8.6	Initialisation file: operator switch parameters . . . . .	181
8.7	Standard EA - gateway constrained operator (switched) fitness plot . .	182
8.8	Gateway constrained - standard EA operator (switched) fitness plot . .	182
8.9	Standard EA (base-line)- Gateway Constrained operator Fitness differ- ence plot . . . . .	185
8.10	Standard EA (base-line)- best hybrid operator Fitness difference plot . .	185
9.1	Real world abstraction . . . . .	194
9.2	Initialisation file: Original input data-set error parameters . . . . .	201
9.3	Initialisation file: Multiple input file parameters . . . . .	207
A.1	MICA2DOT Smart Dust mote Circuit board view . . . . .	213
A.2	MICA2 (z) Smart Dust mote (dimensions are in inches) . . . . .	215
A.3	MIB510 RS232 Serial Interface . . . . .	216
A.4	MIB520 Universal Serial Bus Interface . . . . .	217
A.5	MIB600 Ethernet Interface Module . . . . .	218
A.6	Crossbow Stargate . . . . .	219
A.7	Stargate functional diagram . . . . .	220
A.8	Stargate dimensions . . . . .	221
B.1	Simple Smart Dust based Radio Sonde block diagram . . . . .	224
B.2	Simple Smart Dust based Radio Sonde block diagram . . . . .	224
B.3	Pressure Sensor and Analogue to Digital Converter example . . . . .	225
B.4	Simple smart dust sonde pseudo-code . . . . .	226
B.5	Clustered drop sonde pseudo-code . . . . .	229
D.1	Synchronised Inter-element Packet Format Description . . . . .	257
D.2	Synchronised Inter-element Packet Format . . . . .	258
F.1	Dataset comparison: 10 to 200 elements - Fitness Plots . . . . .	268
F.2	Dataset comparison: 10 to 200 elements - Standard Deviation Plots . .	269

## **LIST OF FIGURES**

---

# List of Tables

4.1	Approximate geometric permutations against cluster size . . . . .	72
4.2	Single point operator . . . . .	83
4.3	n-point Crossover operator . . . . .	83
4.4	Uniform Crossover operator . . . . .	83
4.5	Range Groups and Weighting Factors . . . . .	103
5.1	Random Landscape Search results table . . . . .	118
5.2	Random Progression Search results table . . . . .	122
6.1	Mutation distances . . . . .	130
6.2	Mutation fitness comparison . . . . .	131
6.3	Population size . . . . .	131
6.4	Population size performance fitness comparison . . . . .	132
6.5	Tournament selection percentage and equivalent chromosome count ranges	133
6.6	Tournament selection percentage fitness comparison . . . . .	135
6.7	Selection mode type fitness comparison . . . . .	137
6.8	Mutation operator fitness comparison . . . . .	139
6.9	Mutate/accept unchanged ratio . . . . .	142
6.10	Chromosome acceptance ratio/percentage comparison . . . . .	143
7.1	Stochastic hill-climber results table . . . . .	154
7.2	Standard EA results table . . . . .	159
7.3	PS10, PS20 & PS50 all-run fitness and standard deviation comparison .	160
7.4	Standard EA - Large Dataset . . . . .	166
8.1	Gateway constraint operator EA results table . . . . .	179
8.2	Operator (ordered) results table . . . . .	183
8.3	Operator (ordered) range group results table . . . . .	184

## LIST OF TABLES

---

## Part I

# Introduction, Literature review, and Background



# Chapter 1

## Introduction

In this chapter the motivation for the research subject of the thesis is introduced through the preliminary discussion of the background subject areas that form the relevant aspects pertinent to the overall context. The subject areas are divided into two main areas, the first specifically introduces the *real-world* abstraction and concepts that are the basis of this research.

The second area covers the development of a software modelling system designed to support the application of generalised Evolutionary Algorithm based optimisation methods. In the specific instance used here this system has been configured to resolve the relative positional Euclidean geometries that may exist between a set of notional discrete elements maintained in a 3-dimensional space; the only source of input data available to perform the geometry resolution is a sparse, non-deterministic data-set representative of a non-vectored metric that represents the distance between the discrete elements.

### 1.1 Real world motivation

Of all the current environmental concerns causing world-wide consternation perhaps the greatest is that of global climate change. There have been many research programmes directed at addressing some of the many aspects encompassed by this very active field of research. The one aspect that may be considered as fundamental to all of them is the need for data, to be useful this data must be accurate, timely, available, and above all in a form appropriate to the needs of the research. This requirement forms the primary reason that inspired the research introduced in this thesis; specifically the development of a novel method, or system, to provide an augmented system of near real-time upper-air meteorological synoptic data by building on the concept of an existing environmental monitoring system using modern electronics and software techniques.

The real-world basis for this work is centred around the concept of a simple airborne environmental monitoring device that transmits a synopsis of the monitored data as a structured stream via a simplex wireless communication link to a dedicated receiving

## 1. INTRODUCTION

---

and post-processing station, the latter is usually ground based but could itself also be airborne.

In this section a number of subject areas are introduced that each describe a discrete aspect of how this basic environmental monitoring device can be implemented, and then extended, using alternative generalised computing technology. Finally, how the resultant real-world motivated system can be modelled in a software environment to generate suitable 3-dimensional positional data for a group of the monitoring devices is described.

### 1.1.1 The Radio Sonde airborne environmental monitoring system

Radio Sonde is a generic term for a family of wireless-enabled environmental monitoring devices, there are two main types of Sonde device used in meteorology; these are, the Radio Sonde, (flown aloft by helium filled weather balloon), and the Drop Sonde variant that is deployed, (ie. dropped) from an aeroplane, its descent controlled by a combined parachute/aerial. These Sonde devices are, in effect, small self-contained automatic weather stations that transmit a synopsis of the main meteorological parameters to the receiving station, that may itself be ground based or airborne, generally via a simplex wireless communication link.

The monitoring element of any Sonde monitoring system are untethered, and are effectively '*free-flying*'; the implication of this is that there is now a need to establish the 3-dimensional position of the device in real-time to validate the monitored data.

### 1.1.2 Tracking distant objects

There are established methods that can be used to determine the track of the position of a discrete, free-flying, and autonomous monitoring device in 3-dimensions, and simultaneously to generate the timing information. Successful tracking of any distant object requires a method that can resolve the 3-dimensional position of that object relative to a known frame of reference; RADAR, and the now ubiquitous Global Positioning System (GPS) being two such systems. Although GPS can track many objects simultaneously, it does require that each element is equipped with its own GPS module, tracking individual Sonde devices with RADAR is also quite feasible, the difficulty is in identifying each element uniquely and consistently throughout the entire '*flight*'.

RADAR and GPS are complex systems and will present their own challenges in terms of certain operational limitations and, perhaps more significantly, in their capital cost. It is therefore desirable to establish alternative tracking methods that limit the dependency on such systems; although it may not be possible to dispense with them entirely as the external frame of reference is ultimately still a necessary requirement.

The objective of minimising the dependence of external referencing systems will re-

---

quire an alternative and independent method of determining the 3-dimensional positional data of distant objects; this forms the fundamental core of the research work undertaken for this thesis.

### 1.1.3 The ‘Smart Dust’ system

Recent developments in micro-electronics has resulted in a practical system of general purpose micro-controller based sensor platforms, colloquially described as *Smart Dust*. These devices are, in effect, miniature generalised computers that are equipped with a range of digital and analogue I/O ports, onboard Digital to Analogue Converters, but most significantly, they also have a built-in duplex wireless system; this enables them to communicate between themselves, and through purpose-designed *docking* devices, to interface with existing computer based networks.

### 1.1.4 A Smart Dust based Radio Sonde

A single Smart Dust device can be configured, or programmed, to function in a manner analogous to a generalised Radio Sonde monitoring device. The main additional requirements are a suite of appropriate sensors and dedicated software program to perform the Sonde-derived functionality; specifically to interrogate the sensor suite, store, assemble the data packet, and then transmit the resultant data as a stream on a cyclical or on-demand (polled) basis.

With the addition of a second Smart Dust device configured to function as receiver and logging module a system has been specified that can perform the task of an existing *standard* Radio Sonde, such as those used in meteorology.

### 1.1.5 Multiple element Radio Sonde

A logical step from this is to specify a cluster of Smart Dust devices, each configured as a discrete Radio Sonde sensor device. As a result of the diminutive form factor of most Smart Dust devices it is quite feasible to deploy a number of them simultaneously as a co-operating cluster. In this mode each discrete Sonde element will monitor the environmental parameters from its own *local track*, this produces a set of discrete synoptic data streams. These have the shared characteristic that they are related in both time and near physical location, as a result these data streams can be combined such that the overall environment is being monitored in 3-dimensions; the volume of space being the area described by the physical dispersal of the Smart Dust Radio Sonde devices and their resultant *flight path*.

Although the multiple data streams produced by a multiple element Sonde system are related in both time and near-proximity, it is still necessary to determine a verifiable spatio-temporal context before any of the monitored meteorological data, whether as a discrete entity or in combination, can be considered as validated.

## 1. INTRODUCTION

---

### 1.1.5.1 Generating the required spatio-temporal context

Although the spatio-temporal context could be provided by systems like RADAR and GPS as discussed above, it is one of the fundamental themes of the research work of this thesis to develop alternatives to these external systems. A possible alternative makes use of additional source of data that is automatically generated by the duplex wireless communication system during normal data transmissions by every discrete Radio Sonde element in cluster. The method introduced here requires a group of at least three discrete monitoring devices to enable the conditions for an implicit triangulation function to exist between them.

Every time a Radio Sonde element transmits a synoptic data message, all other Sonde elements, which includes the ground station, will receive this transmission. The duplex wireless system on the Smart Dust devices detects the received signal power level and automatically generates the Receive Signal Strength Indicator (RSSI) parameter. The level of this RSSI value does, in turn, reflect the magnitude of the distance between transmitter and receiver. If all such signal strength values are recorded and combined into a spatio-temporal data-set consisting of the signal strength, the reception time and the element source/destination pairing a snapshot of the overall relative distances between all elements has been created. Further processing may reveal the 3-dimensional geometry encoded within.

### 1.1.6 The software modelling abstraction

The method employed to determine any potential 3-dimensional geometry encoded within the spatio-temporal RSSI input data-set is through the development of a software based system that models a simple airspace-type environment. This software environment ‘*encapsulates*’ the 3-dimensional airspace in which the cluster of unrestrained discrete Smart Dust elements, (configured as Drop Sonde devices), are represented as object oriented data structures.

In terms of the number of possible candidate solutions for any given RSSI based *geometry*, the complexity can be readily demonstrated as being at least Non-deterministic Polynomial Hard (NP-Hard) by (Garey & Johnson, 1980) [44], and in the presentation by Kleinberg & Tardos, 2008 [73]. As a result any exhaustive search algorithm is likely to be completely untenable; for there to be any likelihood of being able to generate candidate solutions in a reasonably useful time-scale. The model has been designed to directly support generic optimisation algorithms implemented using a generalised, virtual framework interface.

The initial algorithms implemented were completely random in that they did not apply any sort of heuristic control, these cannot really be defined as optimisation algorithms, they proved useful as the results from these supported the notion that the overall complexity of the problem space is not trivial.

---

The first *real* optimisation algorithm to be implemented was a relatively simple ‘*stochastic Hill-climber*’, the results generated by this algorithm, at least for the simpler problem spaces, proved to be better than anticipated. All of the remaining optimisation algorithms implemented through the generalised virtual framework interface were various forms of Evolutionary Algorithm. The definition of these implies a wide range of algorithm control parameters and operator types to enable the fine-tuning to match the requirements of any given problem space; in addition the operator mechanisms can be used to implement additional non-EA heuristics.

The software model is central to the majority of the work of the thesis, it is described in some detail in the problem specification chapter 4.

## 1.2 Principal contributions

The principal contributions of this thesis include the development of a number of new computational techniques, together with the main research findings of the work. In the terms of the main research findings the principal contributions are as follows:

- **A virtual base-class optimisation software modelling system.**  
This development of an object oriented software Virtual Base-class Framework (VBF) modelling sub-system designed to encapsulate the salient requirements for the generalisation of optimisation algorithms. This software system provides the required iterative run-time sub-structure that automatically makes use of the chromosome, mutation & crossover operators, input & output data-set definition objects, run-time multiple run data collation and objective fitness function objects ‘*specifically designed*’ to model a particular problem space implementation.
- **Search space simplification through identified defining characteristics.**  
It is necessary to define the limiting characteristics that describe a particular problem space in order to adequately represent it in a computer environment, in this thesis this process has been extended to further identify means by which the size of the search space may be reduced through the elimination of ‘*impossible*’ parameter permutations. This strategy has resulted in a reduction in the run-time required to complete a pre-set number of iterations or to reach an objective fitness trigger level to stop the processing.
- **Heuristic operators.**  
One of the practical methods by which some of the search space simplifications has been implemented, a new set of mutation and crossover operators have been directly derived from the ‘*standard*’ set of operators, which have been directly derived from the relevant VBF classes. The (VBF enforced) class hierarchy allows the simple switch from one form of an operator to another (within the same

## 1. INTRODUCTION

---

hierarchy) which directly enables the easy application from a ‘*standard*’ algorithm to one using heuristics at any point in the run if the run-time conditions (ie. if stagnation is detected) recommend it.

- **Run-time switchable hybrid algorithms.**

The work of the thesis has shown that the basic algorithm types used in the thesis frequently exhibit different levels of ‘*ability to resolve candidate solutions*’ relative to each other, and at different times during the comparative run-time process. As all implemented algorithmic run-time objects share a common software framework (through the VBF), they can be instantly switched, at any point in the run, between algorithms as often as required, (ie. from Stochastic Hill-climber to Evolutionary Algorithm). This mechanism can be extended to switch any set of VBF entities during run-time as the hybrid algorithm demands, for example, the aforementioned operator switching, or changing the active fitness function to allow for objective modification during the evolution of an algorithm run.

- **A run-time convergence rate ‘quality’ metric.**

This concept builds upon the concept of the objective fitness function as a performance measure for a convergent optimisation algorithm. This requires a data-set of objective fitness data points generated through the normal iterative stages of the algorithm to generate additional run-time performance metrics, there are two distinct forms that address distinct aspects of the run-time performance. The first involves the performance across a number of discrete runs on the same problem space, input data-set and run-time parameter settings; this takes the form of a ‘*Range group hits metric*’ that returns a string of numbers that indicate how many of the runs were in pre-set ‘*Range-groups*’ (RG) that cover a range of outcomes from perfect to unacceptable objective fitness.

The second form uses a range of weighting factors applied to the RG values obtained, where a perfect result has been configured to result in the quality factor of zero for a minimising objective fitness function, or some maximum value maximising objective fitness function. When used in conjunction with the terminal objective fitness values and the standard deviation from across all runs, the result produced reflects the overall performance of the algorithm throughout all runs and not just the terminal objective fitness value. This provides a useful means by which performance ‘*bottlenecks*’ may be identified.

- **A ‘*Smart Dust*’ clustered Drop Sonde**

The identification of a novel application for the ‘*Smart Dust*’ sensor platform system as an augmented multi-element clustered Radio Drop Sonde hardware and software system; this is not only novel in the method of implementation, but also in the range of 3-dimensional data types that will be produced during real-world deployment of such a system.

---

## 1.3 Chapter list précis

### 1.3.1 Chapter 2 : The literature review

This chapter reviews some of the main subject matter felt most pertinent to work of the thesis using literature from the body of work currently published. The subjects covered include, environmental monitoring using Smart Dust, localisation, RSSI, and optimisation algorithms based on the evolutionary method.

### 1.3.2 Chapter 3 : Background

The objective of this chapter is the simple introduction to the main subject areas relevant to the hardware and software abstraction that underpins the work of the thesis. The subjects cover a diverse range of material, but all are relevant to some aspect of the overall problem definition.

### 1.3.3 Chapter 4 : Software model design

This chapter introduces and describes the research subject addressed in this thesis. The core of the chapter is based on a working abstraction of an existing real-world environmental monitoring system that is subsequently translated into a software design abstraction. The software implements a ‘*Virtual Base-class Framework*’, this is one of the major themes of the work of the thesis; *all* of the algorithms implemented in the run-time simulations have been based on this software framework. The concept behind the VBF is, in principle, straightforward, in practise it can be a little more complex. The overall design idiom is to provide a generalised object-oriented software sub-system that, as far as is practicable, defines the principal defining characteristics of a generalised optimisation algorithm, in so doing enables the derivation of related optimisation algorithms, thereby ‘*automating*’ the process of software development to some extent.

In addition to the high-level description of the software this chapter describes the salient characteristics of the real-world abstraction of the Radio Sonde system that largely defines the problem search space: this is directly reflected in the design of the computer representation. The functionality of the computer representation elements are also described to illustrate the encoded algorithms, and the significant run-time and compilation data and control inter-connections.

### 1.3.4 Chapter 5 : Run-time base-line using non-optimised algorithms

In this chapter the objective is to describe the implementation of simple, non-optimised, software models directly derived from the Virtual Base-class Framework (VBF). The additional objective is to demonstrate that the run-time software based on the VBF does not itself impose any underlying algorithm type effect into the system. More specifically to demonstrate that the VBF simply provides the means by which an optimisation

## 1. INTRODUCTION

---

algorithm may be implemented, but that it does not noticeably influence the generation of candidate solutions; the code implemented by the derived algorithm classes alone must provide this *intelligence*. To meet this requirement a set of simple algorithms are implemented that are essentially random in operation.

### 1.3.5 Chapter 6 : Establishing optimal algorithm parameter settings

The set of run-time parameters normally associated with Evolutionary Algorithms (EA), such as crossover, selection, recombination, mutation rate, and population size, frequently have a strong impact on the overall performance of an EA. This chapter investigates a number of the more significant parameters and establishes the best setting for each. These have all been derived through strictly empirical means to establish as unequivocally as possible the very best setting. To achieve this each discrete parameter is made the sole subject of a dedicated EA simulation run. The value is based on the average result for all element input data-set Problem Size (PS) sub-sets, (PS10 to PS200), equivalent to the elements in a *'notional'* clustered Drop Sonde.

### 1.3.6 Chapter 7 : Optimisation algorithm simulations

The main objective for this chapter is to implement some simple optimisation algorithm based simulations using the set of Standard Derived Classes (SDC) that have been directly derived from the Virtual Base-class Framework (VBF) as described in chapter 4 and to assess their relative performance when solving the 3-dimensional geometric reconstruction problem. The first part of the chapter defines two optimisation algorithms based on a straightforward level of *intelligent direction*. The application of heuristic, or evolutionary, selection pressure will generate a set of candidate solutions that, whilst still largely defined as stochastic, demonstrate the effect that the selection of the strongest solution can have on convergence.

### 1.3.7 Chapter 8 : Heuristic and hybrid operators

This chapter explores some of the possibilities of developing heuristics that can enhance the overall performance of the reconstruction model by making use of certain characteristics implicit in the design of the model and the real-world abstraction. There are two main themes, the first is the development of a *The Gateway Constraint Operator* that significantly restricts the number of positions a discrete element can occupy in the modelled 3-dimensional zone, thereby greatly reducing the problem search space. The second theme concerns the development of hybrid algorithms based on the *standard* EA operators and the newly developed constraint operator.

### 1.3.8 Chapter 9 : Conclusions and further work

The main objective of this chapter is to summarise the work that has remains incomplete, specifically with reference to dealing with the errors in the input data-set. This covers a number of subject areas, these are mainly concerned with making the problem

---

search space ‘*smaller*’ with the intention of improving the performance of the model in the time taken to resolve a geometry and the accuracy, repeatability and confidence that can be placed on the ‘*correctness*’ of the geometry. The problem space is divided into smaller sections parts that can be easily defined, the benefit of this is that the effects of errors will be locally defined and therefore can be processed using parameter settings that are also locally defined. By this process of local quantification and resolution each section can be expected to exhibit a reduced effect on the overall global fitness of those candidate solutions. This is a prime area for more research into *heuristic operators*, such as those introduced in chapter 8. The second objective is the summary of possible areas for future research work in this area.

### 1.3.9 Appendices

- Appendix A : Crossbow Technology™ Smart Dust system.
- Appendix B : Smart Dust as an automatic weather station.
- Appendix C : The software model initialisation file.
- Appendix D : The software model input data-set file.
- Appendix E : Example software model results file.
- Appendix F : RSSI input data-set neutrality test.

## 1. INTRODUCTION

---

## Chapter 2

# Literature Review

In this chapter the objective is to review some of the more relevant aspects that encompass the research area. The whole subject area could include themes as diverse as environmental monitoring, electronic computer processor wireless hardware, optimisation algorithms, localisation, or software modelling. The introduction and background chapters (1 & 3) describe the work of this thesis in the context of environmental monitoring and localisation in a practical sense. The subjects covered in this review have been selected to include the subjects that most directly support the material in the thesis, which are as follows.

- Environmental monitoring using Smart Dust
- Localisation and Radio Signal Strength Indication
- An overview of ant colony and swarm optimisation
- Optimisation algorithms based on the evolutionary method

### 2.1 Environmental monitoring using Smart Dust

As one of the main themes of the thesis this particular usage of Smart Dust (SD) subject is obviously one of great significance, by definition the use of a general-purpose, micro-processor based, wireless linked environmental monitoring system will find a wide range of potential practical applications, for example:

- Meteorological monitoring
- Agricultural monitoring
- Earthquake monitoring
- Habitat monitoring
- Machinery monitoring

## 2. LITERATURE REVIEW

---

Whilst there is always the potential for many other application areas, the intention here is to simply highlight some of the more common applications. Examples of practical deployment for meteorological monitoring is surprisingly uncommon as a general application; the likely reason for this being the many existing bespoke monitoring systems that have been in use for many years. As a relatively recent innovation SD based systems can quite reasonably be expected to feature in future developments, perhaps as direct replacements for these existing meteorological monitoring and forecasting systems, in global climate and meteorological research, or where additional functionality results directly from the characteristics of the generic SD system.

Global climate and meteorological research is already represented in the Drop Sonde (DS) research field with a number of ongoing research programmes. There are some notable examples of this in global climate and meteorological research that are using DS type devices, currently these devices are all bespoke but it is probable that they could be implemented using SD technology with little technical difficulty.

The first field of research concerns ‘*predictability research*’ into upper-air phenomena by **The University Corporation for Atmospheric Research** (UCAR) organisation [117]. The fundamental DS type device being used is currently a development of the basic **Drift Sonde** [118], to be introduced in section 3.4.6. The aim of this research programme is to provide a longer-term monitoring facility into meteorological phenomena that would not otherwise be readily observable using methods such as terrestrial observation satellite systems. The authors of the work, (Cole et al, 2005) [21], state in this literature that they have an overall research aim that ‘*predictability research suggests that additional in-situ measurements in sensitive regions will improve predictions of high-impact weather events, these regions are often cloudy*’. By the facility of a number of SD type devices remaining in the ‘*area of interest*’ for days or weeks at a time can result in the local, prevailing, meteorological conditions being studied continuously through the generation of large amounts of spatio-temporal data. This has a number of pertinent parallels with the general concept being explored in the work of the thesis. There are two further research programmes that have benefited directly from the **UCAR** research as they are related in both the type of research being undertaken and the means and techniques employed. These are **THE Observing system Research and Predictability EXperiment** (THORPEX) [115] and the **African Monsoon Multidisciplinary Analysis** (AMMA) by (Redelsperger et al, 2006) [99].

A strong theme in the related literature seems to be the number of papers that reflect the current ‘*state-of-the-art*’ of the wireless sensor networks (WSN) technology field through surveys of the technology and how it may be used. (Vieira et al, 2003) [129] stated that the SD system has the potential to provide a relatively cheap means for a system with the potential to be disseminated in large quantities over a large area. SD devices, being largely autonomous, can be easily configured to provide a collaborative network of pervasive distributed sensor elements and then left in-situ for an extended

---

period. The conclusion of this survey was that there are some *'architectural challenges'* that must be addressed prior to the deployment of any practical system, these are the expected limitations imposed by the available computational power, energy availability and consumption, and sensing capabilities.

The survey of SD based Wireless Sensor Networks carried out by (Khemapech et al, 2005) [72] considered both military applications and the more recent civil applications. The latter usage becoming more practical due to further miniaturisation of the devices and subsequent reduction in production costs. In operational terms the paper concluded that the main drawbacks in using SD based networks were both hardware and software based, and could be placed into two main categories. The first is the energy constraint imposed by the remote nature of many of the sensor network deployments, in many such applications the use of battery power is made necessary due to the lack of main electrical power; this would also preclude the use of rechargeable batteries, although the suggestion is made by (Feng et al, 2003) [39] that solar panels could be used. Vibration sourced energy has been suggested by (Manjoo, 2001) [83] although it is not really considered to be an entirely feasible method as the energy level and the constancy of supply could not be reliably assured.

The second significant conclusion concerns the reliability and effectiveness of the communication protocols used by the wireless sub-system. Traffic and data congestion in sensor networks resulted in particularly deleterious effects on *'end-to-end'* data transport with congestion control required to achieve satisfactory data transfer, (Wan et al, 2003) [133] stating that there was a strong dependence upon a network routing protocol.

The survey work performed by (Garcia-hernandez et al, 2007) [43] focused mainly on the technologies, standards (software and hardware) and the applications frequently associated with Wireless Sensor Networks (WSN), with particular emphasis on the need to satisfy the constraints such as *'fault tolerance, scalability, cost, hardware, topology change, environment, and power consumption'*.

The conclusion from this survey work was that WSN offer the potential for significant long-term research potential, but that they may pose as many new systematic challenges as the systems that they are used to implement. In particular the probability that the optimisation problem they may represent will be complex and wide-ranging, these will include some subject areas relevant to the work of this thesis; specifically, localisation, techniques for the physical deployment of the sensor elements, and subsequent tracking. One conclusion of particular interest concerns the use of multiple sensors types in an integrated package, such as would be found in a Drop Sonde type of device proposed here.

The survey work carried out by (Akyildiz et al, 2002) [1; 2] & (Potdar et al, 2009) [97] reached broadly similar conclusions. In common with the previous surveys the

## 2. LITERATURE REVIEW

---

limitations of the constraints introduced by performance factors including fault tolerance, scalability, initial and on-going costs, the deployment environment, and of most significance, the overall power consumption.

Potdar et al also made the comment that the processing capability of these devices is limited and that, as a result, for the majority of WSN configurations most of the actual data processing will be carried by computer systems external to the WSN element cluster as post-processing events. This is both logical and practical as the available processing power and run-time memory and storage on the SD devices will severely limit what can be achieved, post-processing a data-set ensures many more options will be possible.

### 2.2 Localisation and Radio Signal Strength Indication

The use of the Radio Signal Strength Indication (RSSI) parameter as the prime source of localisation data is widespread in both quantity and in the range of usage subjects covered in the research. As a simple piece of raw distance magnitude data RSSI can be interpreted in many different ways according to the needs and requirements of the research.

For this to be a ‘safe’ approach to using RSSI in the manner required it needs to be clearly understood in terms of the potential information that it may provide and, in particular, in the restrictions and caveats implicit in the data as a direct result of how it is generated.

As a result there are many research projects that endeavour to establish an understanding of the limitations imposed on RSSI as a reliable distance metric. There are a number of these that include many types of environmental considerations, variations in the transmitter & receiver electronics, or signal strength variations caused by something as simple as the relative orientation of the aerials.

A question of some relevance has been posed by (Heurtefeux and Valois, 2012) [58], ‘*Is RSSI a Good Choice for Localisation in Wireless Sensor Network ?*’. In this paper the general conclusion was that the localisation estimation using RSSI is a relatively poor representation of the distance magnitude between a transmitter and receiver pairing, the work involved a direct comparison between the distances obtained by applied RSSI and simple Euclidean calculation was an order of magnitude worse, specifically metres to centimetres difference in favour of Euclid.

Most notably this paper employed sensor networks of up to 250 elements in several different environments and topologies, and using a number of different wireless ‘chips’, (Texas Instrument™ CC1101 & CC2420), operating on their respective frequencies, thus effectively removing any significant bias that the hardware itself may have intro-

---

duced. The work also concluded that to obtain a stable and accurate RSSI distance magnitude representation, the affects of a number of environmental criteria would, in the ideal situation be taken into consideration addressed. These can be summarised as being able to measure RSSI across several different frequencies (frequency hopping/spread spectrum)<sup>1</sup>, to be able to take a number of RSSI measurements to average out variations, calibrated wireless ‘chips’, and finally to use good quality antenna, ideally as omni-directional as possible.

Overall it was concluded that it is highly desirable, if not absolutely necessary, to be able to minimise general interference and dynamic environmental changes. Whilst they supposed that these constraints are not compatible with a dynamic wireless enabled sensor network deployment it is suggested that the usage as proposed in this thesis would be less affected by such limitations as the upper-air region is largely devoid of excessive encumbrances.

One of the main problems with RSSI that imposes severe limitations on its dependability as a measure of distance between two, or more, discrete wireless elements is the variability inherent in repeated transmissions. This aspect was researched by (Fang et al, 2010) [38] with particular emphasis on the variability of RSSI and its characterisation and calibration. This paper concluded that RSSI can work well as a means of localisation in the ideal, open, outdoor environments where the restrictions such as walls are minimal. The caveat as always was ‘*RSSI remains an extremely challenging task because of the effect of reflecting and attenuating objects in the environment*’.

The study of the characteristics of the RSSI signal by (Wu et al, 2008) [141] followed a similar approach that dynamic changes in the environment plays the greatest role in affecting RSSI signals, in addition they found that the regularity and repeatability of RSSI signals could not be established with any certainty. They also concluded that multipath fading had a strong deleterious effect. a finding equally supported by (Pu et al, 2008) [19].

Research by (Huang, 2009) [65] found that from a purely hardware viewpoint variations in relative aerial (antenna) polarisation or orientation can have a notable and indeterminate influence on the RSSI to distance magnitude relationship. Although the technical documentation for the wireless ‘chip’ (the ChipCon™ CC2430), stated a nominally linear ‘*typical RSSI value to input power*’ relationship (when logarithmically normalised), the actual real-world performance displayed a greater level of variability from the largely linear relationship in the technical documentation for the wireless ‘chip’. The conclusion that can be drawn from this research is that antenna orientation has a strong influence on the signal strength to distance relationship that introduces non-linear errors that cannot be predicted without prior knowledge of the relative ori-

---

<sup>1</sup>These are advanced techniques that require an adaptive algorithm and may itself introduce RSSI data drop-outs if synchronisation between transmitter and all potential receivers is not maintained.

## 2. LITERATURE REVIEW

---

entation.

(Wadhwa et al, 2009) [130] also found that the antenna orientation is an important factor that can significantly affect the RSSI to distance magnitude performance, they found that even if the corresponding antennae were set in a fixed perpendicular relationship to each other the RSSI to distance magnitude could still be adversely affected. Their final conclusion was that there *'is a direct impact of antenna orientation on the received signal strength and hence the performance of the network'*. It may be surmised from these results that an omni-directional may be the only effective solution to this problem; although it must be stated that such a device is largely idealised, there are antenna forms that can quite closely approximate a uniform electro-magnetic field with the caveat that there will be a resultant reduction in field strength density with increasing distance when compared with a dipole type of aerial.

Further work that focused on the more practical aspects of determining the geometry of wireless enabled elements. (Zheng et al, 2010) [143], (Reichenbach & Timmermann, 2006) [101], & (Daiya et al, 2011) [24] focused on the concept of triangle and centroid location algorithms. Zheng et al concluded that their simulation results indicate that the combined triangle and centroid location algorithm has a better localisation accuracy than the traditional RSSI localisation algorithm used alone. In the context of the work of this thesis this conclusion can be largely supported as the triangle/centroid data used in their work is somewhat implicit in the concept of a 3-dimensional geometry *'encoded'* in the RSSI data-set as used here; the confirmation of this is something being considered as possible further work.

In the case of Reichenbach & Timmermann a *'Weighted Centroid Localisation algorithm'* was employed in combination with RSSI data to good effect without the need to resort to any form of centralised, server based, location maps. The fundamental work that characterises this work is the analysis of the RSSI to distance curves that defined boundaries to the quality of the data.

Calibration of the hardware involved in the wireless connection is one way of reducing the effects of one source of RSSI induced errors in localisation. To do this for the *'chip'* in every wireless element is quite prohibitive in practical terms, to help overcome this limitation (Barsocchi et al, 2009) [9] proposed a method of *'Virtual Calibration'*. The usual process of determining a calibration map is through the a-priori calibration of the propagation model (called fingerprinting), this requires extensive pre-processing to create the required RSSI distance magnitude relationship map. By exploiting the RSSI values between pairs of elements an average can be taken as the two values should ideally be identical. This method of pre-processing will lead to a more robust RSSI to distance value, this method is incorporated in the software model used in this thesis when it is in *'error-correction mode'*, the confidence of this data is further enhanced by the addition of more element-pair values.

---

An alternative method of calibration is the '*Heuristic Environmental Consideration Over Positioning System*' (HECOPS). This is a distributed location algorithm for wireless sensor networks which operates by every element estimating its physical location during the process of interacting with the other elements. This decentralised location system approach devised by (Reghelin et al, 2006) [100] uses a number of reference elements that have a defined '*knowledge*' of their location, (perhaps through GPS), to act as '*anchor*' elements. In general the number of such elements is approximately 10 to 15% of the total number of elements; this represents a significant cost factor, as well as the technical challenges presented by GPS, most specifically that GPS needs a '*view of the horizon*' to function correctly. Nevertheless the result presented indicate that the algorithm is effective in providing localisation data for the group of elements.

(Pires et al, 2008) [96] also described and evaluated the HECOPS localisation algorithm through applying a number of simulations. The observation was made that it is quite possible to use an RSSI-based location algorithm with the proviso that sufficient number of anchor elements were employed to support and augment the active localisation algorithm. The conclusion was that although RSSI is not generally considered to be reliable as a method of distance estimation, (mainly due to the instability and susceptibility to noise and interference), it was found that the combination of calibration, sufficient '*anchor*' elements, and some applied heuristics, could overcome these inherent limitations and produce reliable results.

There are many research projects that require some form of RSSI pre-processing technique to establish the relative co-location for a number of wireless linked elements. They can be sub-divided into a number of groups that cover slightly different aspects of the localisation problem, they share the same underlying requirement to apply some form of mapping technique in order to establish relative locations. These can range from '*multi-robot motion planning*' as developed by (Guo & Parker, 2002) [52], '*collaborative mapping and exploration*' by (Burgard et al, 2000) [15], '*formation control*' as a result of the work by (Fierro et al, 2001) [40], '*robotic localisation*' by (Roumeliotis & Bekey, 2002) [108] and general '*communications mapping*' techniques through the work of (Hsieh et al, 2004) [63] (Xiang et al, 2005) [142].

All of these research projects put great reliance on the fact that the wireless elements all calculated their relative positions with reference to a common coordinate system, the details of which was known to all elements at run-time. The source of the common coordinate system is a perennial problem as it has a cost, (in terms of processing time, coherency, and storage), the absence of such a system however implies that the only '*practical*' method that remains for a wireless element to '*localise*' itself is by utilising one of the embedded indicators like the RSSI sub-system.

The conclusion based on the review of the research work included here is that the use

## 2. LITERATURE REVIEW

---

of RSSI as a means by which to establish a relative geometric co-location of wireless enabled elements is entirely feasible if suitable precautions are taken to limit the effects of environmental conditions and variations imposed by the electronics and antennae. The localisation results can always be expected to exhibit a level of indeterminacy such as the inherent unreliability of the RSSI technique when used in the manner described here. What is noteworthy here is that none of the research produced localisation results that were completely accurate, (or even that close), and perhaps of greater significance, demonstrated reliability and repeatability in producing those results.

### 2.3 An overview of ant colony and swarm optimisation

The principles that characterise the ‘*Ant colony*’ and ‘*Swarm*’ optimisation techniques bear some resemblance to the fundamental principle behind the work of this thesis. They share the same fundamental and underlying concept of multiple ‘*entities*’ existing in a defined ‘*dimensioned space*’, and that they have some level of autonomy, or ‘*life*’ associated with them, either as individuals, or as a swarm (or cluster). These techniques have also been used to solve a similar range of problems as for other optimisation techniques, such as evolutionary algorithms; these factors make it appropriate for inclusion here.

The general theory of Ant Colony Optimisation (ACO) has been researched through a survey of existing publications by (Dorigo & Blum, 2005) [27]. In this paper the focus was largely on understanding the ‘*meta-heuristics*’ required to deepen the understanding of why and how a particular method or technique functions in order to improve the applicability of the ACO technique. The general conclusion drawn was that although the research field concerned with experimental ACO work is currently flourishing, greater understanding of the underlying methodology can only help the applicability of ACO still further.

In the guest editorial special section on ant colony optimisation in the IEEE Transactions on Evolutionary Computation, the work of (Dorigo & Gambardella, 2002) [30] established that one of the main characteristics underpinning the ACO technique is that of ‘*self-organisation mechanisms*’. They summarized that the use of ACO has been proposed as a novel computational model that could functionally replace some of the more traditional approaches. The method involved uses control, pre-programming, and centralisation in the design to implement a system with a defined level of autonomy, emergent ‘*intelligence*’, and a distributed functionality model.

The range of problem space applications that ACO can be used to provide candidate solutions is probably at least as wide as for any optimisation algorithm type. Examples of this are the traditional ‘*Travelling Salesman*’ problem and, appropriate to the work of this thesis, to the problem of ‘*data routing in packet-switched networks*’ as suggested by (Dorigo & Di Caro, 1999) [28]. In this paper a common framework

---

system has also been applied to define some ACO meta-heuristics that controls the runtime operation of the algorithm. The ‘*Travelling Salesman*’ problem has been further researched through the implementation of an ACO based system by (Dorigo & Gambardella Dorigo, 1997) [29]. The conclusion reached in this research paper is that the framework based approach to ACO research is currently very strong, with the expectation that many combinatorial optimisation problems can be solved using the technique.

Ant colony optimisation techniques have also been used to research routing problems for use in a Mobile Ad-hoc NETWORK (MANET). This is particularly relevant to the work of this thesis as MANET’s are frequently, although not exclusively, constructed using ‘*Smart Dust*’ devices. (Gunes et al, 2002) [51] proposed an Ant-colony-based Routing Algorithm (ARA) as a “highly adaptive, efficient and scalable routing protocol” to address the main challenge in mobile multi-hop ad-hoc networks, a problem that is exacerbated by the dynamic mobility of the discrete elements, (motives, or nodes), in the network.

The concept of Particle Swarm optimisation (PCO) has been defined by (Kennedy & Eberhart, 1995) [71] as being somewhere between genetic algorithms and evolutionary programming as it is dependent on stochastic processes to function. The work takes an unusual approach by modelling successful traits of ‘*adaptive social behaviour*’ in the animal kingdom to solve engineering optimisation problems. This technique is effective due to the overall effect of stochastic factors that allow a more thorough search of the problem space to help ensure that the search process does not get unduly affected by regions of poor fitness.

Another technique somewhat appropriate to the work of the thesis is that of data-clustering using ‘*K-means*’ through the application PSO. In the work by (van der Merwe & Engelbrecht, 2003) [125], the approach is to use a PSO to determine the ‘*centroids*’ of a number of data clusters. The conclusion was that when compared to the performance of ‘*standard*’ K-means clustering the PSO clustering techniques appeared to show great potential with a better level of general convergence and fewer quantisation errors that define the data under evaluation.

The work of (Shi & Eberhart, 1999) [111] established the performance of the PSO through an empirical study to illustrate the advantages and disadvantages of the technique. The work required four different benchmark functions, set with asymmetric initial range parameter settings, to establish the effects these have on the convergence rate to the optimal condition. It was found that the convergence rate using the benchmark functions was always fast, but slowed considerably when approaching the terminal condition. It was also discovered that the PSO may completely fail to find an optimal solution if the problem to be solved is complex, it was concluded that this drawback may be mitigated to some extent by ‘*employing a self-adapting strategy*’ to the parameter settings.

## 2. LITERATURE REVIEW

---

In conclusion, the concept of multiple ‘*entities*’ existing in a defined ‘*dimensioned space*’ as used in ACO systems do differ in one important aspect to the 3-dimensional multiple element Drop Sonde cluster approach used in the work of this thesis. The ‘*entities*’ used in ACO techniques do, in general, function as a ‘*single entity with some degree of common purpose*’, in effect they are all following each other in some form of mode of collective ‘*consciousness*’.

The cluster of Drop Sonde elements, as used for the work of the thesis also exist in a sort of collective, but they have a greater level of autonomy in their operation, one of the few times they are in ‘*single entity*’ mode is when they are communicating between themselves in order to establish a communication protocol and transmission pattern. In addition the Drop Sonde elements have no control over ‘*where they go*’, they are ejected from an aeroplane and are then entirely subject to the forces of nature that directly impinges upon them and therefore will control their ‘*flight*’ pattern. These differences aside there are likely to be many common areas in how the ACO systems and the 3-dimensional Drop Sonde cluster system are represented in the computational environment.

### 2.4 Optimisation algorithms

There are a number of optimisation techniques that have been developed to solve combinatorial problems of many different types and with equally many different real-world applications. One of the most significant underlying characteristics that define this set of algorithms is the application of random effects to the process of creating solutions. For the simpler types, such as ‘*Tabu Search*’ there are a series of tutorials by (Glover, 1989, 1990, 1993) [45; 46; 47; 48], in the case of the ‘*Stochastic Hill-Climber*’ (SHC) an introductory tutorial is provided by (Tuson, 1998) [116]. The work by (Dunn, 1998) [33] compares the performance of an SHC with that of an Evolutionary Algorithm on the same problem space, (Wang et al, 2009) [135] describe an SHC with an element of built-in crossover to create a hybrid ‘*memetic*’ algorithm.

Whilst there are other techniques that may provide a finer degree of control, (such as ‘*Simulated Annealing*’ to slow the mutation process when a solution is considered close, and in the application of ‘*anti-stagnation*’ techniques where a disruptive, ‘*apocalyptic*’, level intervention is applied to produce large, almost uncontrolled mutations to avoid becoming trapped on local solutions that may be less than optimal; in either case the overall modus-operandi is always strongly stochastic with minimal selection applied in the production of candidate solutions.

The application of an evolutionary process to direct the generation of candidate solutions can be considered as a step above the ‘*Tabu Search*’ and ‘*Stochastic Hill-Climber*’, the result is a class of optimisation technique that still relies heavily on the stochastic

---

approach but with the facility to evolve using basic selection and solution crossover methods to derive new candidate solutions in a system where *survival-of-the-fittest* by (Darwin, 1859) [25] is a predominant factor.

### 2.4.1 Evolutionary Algorithms

The Evolutionary Algorithm (EA) is the primary algorithm used throughout most of the thesis, this type of algorithm may be considered as providing the means to design and implement a generalised system that can generate candidate solutions for a wide range of problem spaces. This may be considered one of the main strengths of the EA, but at the same time it can also be the main source of potential difficulties. It is imperative that the design of the chromosome, objective fitness function, and operators must provide an appropriate, and concisely modelled representation to ensure the best possible performance.

The seminal work that led to the development of the EA is considered by many to be ‘*Genetic algorithms in search, optimisation and machine learning*’ by (Goldberg, 1989) [49], this book laid down many of the founding principles of this type of optimisation algorithm. Another book of major publication significance is ‘*Evolutionary algorithms in theory and practice*’ by (Bäck, 1996) [6], this publication covers many parameter considerations for the four major Evolutionary Algorithm paradigms, Evolution Strategies, Evolutionary Programming, Genetic Programming, and Genetic Algorithms.

The remainder of this section will examine some of the problem spaces that EAs have been used to provide candidate solutions. There are many examples of this including, solutions for the multidimensional knapsack ‘*packing*’ problem researched by (Li et al, 2006) [79], (Chu & Beasley, 1998) [18], (Horowitz et al, 1974) [62] & (Ross & Tsang, 1989) [106]. Facial recognition is an example of another area of research for countering illegal activity, this was researched by (Fujiiwara & Sawai, 1999) [41] by the application of EA to mesh optimisation of 3-dimensional facial images.

One particularly interesting area of research is in the application of ‘*Low-Earth orbit satellites*’, this type of satellite has a significant operational drawback in that they must, by definition, drop below the horizon, (from a terrestrial view-point), on a frequent basis. The time and frequency that they are visible is dependent on the height of orbit, for example, the Hubble telescope is visible for only 12.8 minutes once every 96 minutes. Therefore the challenge is to minimise signal coverage blackouts, this led to research to determine the optimal arrangement of relative co-location for a group of satellites to satisfy this requirement. EA have been applied to directly address this optimisation problem with good results by (Williams et al, 2001) [139] & (Jilla, 2002) [68]. Low orbit satellites are far less costly than their geo-stationary counterparts, by producing optimal low-Earth orbit patterns provides a valuable cost saving.

Another area of great significance in the modern world, (and indirectly to the work

## 2. LITERATURE REVIEW

---

of this thesis), is in the design of modern integrated circuit electronic circuitry. This application has become increasingly complex, where even relatively ‘*simple*’ electronic circuits can routinely consist of many millions of electronic elements, such that the original stipulation in 1965 by ‘*Moore’s law*’ by (Moore, 1998) [86] ‘*that the number of transistors on a chip will double approximately every two years*’ has been recently exceeded, (a recent AMD™ CPU has 1.2 Billion transistors). With this level of complexity computer assistance in the design of circuit layouts has become almost mandatory. The field of ‘*Very-Large-Scale-Integration*’ (VLSI) therefore presents an ideal problem space for the application of an optimisation algorithm approach where there are a significant number of possible permutations in the design of the circuit layout.

One of the main challenges with implementing VLSI concerns the physical positioning of the electronic components and the subsequent track connections between them, this task must be completed on the resultant circuit board layout to ensure the efficient operation of the circuit to maximise performance; this is an example of the Partitioning problem researched by (Cohoon et al, 1991) [20] & (Drechsler, 1999) [31]. Another aspect of this development process researched by (Koziel & Szczesniak, 1999) [75], is to address other significant design issues pertinent to electronic systems, such as minimising heat transfer between sensitive components that are positionally adjacent and, of increasing importance with modern communications systems, of avoiding electromagnetic field interference.

VLSI can be considered as an ideal search problem, with the objective of generating the best layout within the constraints of the electronics, this makes it a subject area that EA are ideally suited, a conclusion reached by (Lienig, 1997) [80; 81] (Cohoon et al, 1991) [20], (Shahookar & Mazumder, 1991), [110] & (Koziel & Szczesniak, 2003) [76].

### 2.4.2 Evolution Strategies

Evolution Strategies (ES), are a sub-class of the Evolutionary Algorithm (EA) optimisation methodology paradigm, although they actually pre-date EA by at least a decade. There are significant similarities in their shared underlying methodology of nature-inspired combinatorial search and optimisation. In common with the EA methodology they implement mutation, recombination, and selection using defined methods iteratively applied to a population of chromosomes that represent, or model, the specific problem space; the resultant candidate solutions are then assessed for objective fitness for purpose in the context of the problem space.

One significant area that ES and EA differ is in the apparent contradiction of the concepts indicated by the words ‘*evolution*’ & ‘*strategy*’ being used in the name. Any suggestion of a strategy, or applied heuristic, indicates a level of direction that may represent a level of evolutionary pressure on the evolutionary process, unless the heuristics are carefully selected and implemented. Under this stipulation the ES may be reasonably considered as an EA with heuristic operators.

---

Although ES may be used for many classes of problem space in a broadly similar manner to that of the EA, they were not originally devised to compute minima or maxima real-number values from static (objective fitness) functions in the manner of an EA. The original intention was to automate the design and analysis phases of consecutive, comparative experiments. The general approach was to optimise a set of discrete variables that define a problem to be solved, (Beyer & Schwefel, 2002) [10] describe an ES based experiment where the parameters that define a 2D ‘plate’ in a turbulent air flow were efficiently determined using a simple randomized ES heuristic; the ES method outperformed other more traditional mathematical based modelling methods.

The use of ES as a strategy based optimisation method can be readily identified in the work of (Pereira-Neto et al, 2005) [95] & (Varela et al, 2003) [98]. These papers sought to establish a set of parameters to ‘*solve an economic dispatch problem*’, and to reduce ‘*scheduling problems that have bottlenecks*’ respectively. The problems solved in both research papers required that a set of parameters pertinent to each problem space were determined to enable the efficient scheduling and dispatch process from within systems that have a set of non-linear restrictions imposed on them by their respective environments. To fulfil a broadly similar requirement (Roubos et al, 1999) [107] proposed a system to ‘*calculate optimal control policies for bio-reactors*’; the system to determine these parameters used an ES that was actually based on a Genetic Algorithm (GA), which supports the notion of the close relationship that exists within this group of optimisation algorithms based on evolutionary principles.

Another, rather more topical uses for ES have been in the work of (Mester et al, 2003) [84] & (Vidyardhi et al, 2005) [128]. The former research work concerned the construction of large-scale genetic maps of 50 to 400 genetic markers, the work combined the travelling salesman approach with multi-linked ‘*loci*’ to create the maps of inter-connections of all possible orders, ‘*amounting to  $n!/2$  for  $n$  loci; hence it is considered an NP-hard problem*’. In the latter research paper, the work involved optimising the channel assignment in wireless mobile networks in the work by Vidyardhi et al. In this paper the problem was to make the most efficient use of the limited number of wireless mobile communication channels available in the radio spectrum, something especially necessary in a congested usage area. The primary parameters that were being optimised were to, minimize call blocking, the rate of dropped calls, and to maximize the quality of service. The latter being somewhat subjective, but it is no less useful an objective, even if it is less easy to define.

The work of (Hansen & Ostermeier, 2001) [54] sought to eliminate the stochastic element from the ES algorithm altogether, by a process of ‘*de-randomization*’ and ‘*cumulation*’. They introduced the concept of ‘*self-adaptation*’ where the iterative, (admittedly still random), mutative steps are performed, they are based on the history of mutation events that have proved ‘*worthy*’ in previous iterations. The ‘*cumulation*’ path method

## 2. LITERATURE REVIEW

---

evaluates a series of mutation events rather than a single event to establish selection instead of the more usual stochastic selection techniques of an EA. They found that this approach ‘*preserves all invariance properties against transformations of the search space and of the objective function value*’, put another way the history of the mutation is preserved and used to guide the future direction of the evolution.

In keeping with the concept of the determination of an overall strategy to design and optimum settings for the parameters that define the problem space the work of (Alavi et al, 1981) [3] approached this through the development of a Decision Support System (DSS) to ‘*establish a learning-based, participatory implementation strategy*’. The application problem space was a (simulated) production environment and used to evaluate the performance of the problem solving efficiency of the system. Their conclusions indicated that the DSS implemented by the ES based algorithm out-performed the more traditional techniques if the inter-connections between the design of the DSS, the final implementation and the programmed learning processes are carefully integrated and represent the problem space adequately.

### 2.5 Chapter summary

The literature surrounding this field of research is extraordinarily wide and varied, as a result this chapter contains a small sub-set of subjects that are partly representative of the main subject matter. These can be broadly described as the monitoring the problem space environment (Smart Dust), the input data-set (RSSI), which results in a set of candidate solutions may be generated (optimisation algorithms).

It was perhaps an impossible task to perform an exhaustive review of all the fields of research for which literature exists, the intention was to highlight some aspects of the subject matter areas that are felt to be relevant to the work of the thesis so they can be used as a starting point for further reading.

## Chapter 3

# Background

The objective of this chapter is to introduce the main subject areas relevant to the hardware and software abstraction that underpins the work of the thesis. The subjects cover a diverse range of material, but all have specific relevance to a particular aspect of the overall problem definition.

The intention is to place the research subject abstraction, as far as is reasonably practicable, into a real-world context based on an existing environmental monitoring system. In so doing what may have been a purely academic exercise of developing novel methods of finding optimal 3-dimensional relative positional geometries of a number of notional objects able to freely circulate anywhere in a 3-dimensional space can now be directly linked to this real-world implementation.

### The areas to be discussed

- Defining localisation
- Relative distance identification
- An overview of contemporary range detection
- **Radio Signal Strength Indication (RSSI)**
- An overview of the generic Radio Sonde monitoring system
- An overview of evolutionary algorithms
- An overview of the object oriented paradigm
- Problem space representation in a computational environment

Each of these individual subject areas has directly contributed to the real-world abstraction, and therefore indirectly, to the design of the software system that models the 3-dimensional *zone* into which the discrete objects are mapped, manipulated and then subject to geometric analysis. In so doing this promotes what would otherwise be

### 3. BACKGROUND

---

a purely academic exercise of determining an optimal relative positional 3-dimensional geometry of a number of freely circulating notional objects into one that can now be directly linked to a practical real-world implementation where this concept is fundamental to the operation.

The problem definition, software model design, specification and implementation aspects based on this abstraction are covered in chapter 4. Illustrating the abstraction using this *visualisation* approach has proved to be an extremely useful tool in itself as the real-world abstracted objects can be directly translated into software objects. The software implementation has been designed and built using the object oriented design idiom as defined in this example by (Henderson, 1993) [57] resulting in the model design used as the basis for all simulations in this thesis.

#### 3.1 Defining localisation

The term *Localisation* is used in a wide range of quite different contexts, as a result it has a wide range of possible interpretations that are defined on that context. They have a shared common attribute in that they all refer to the relative, notional, or actual physical position of some feature dependent on that context. Subsequently there are many definitions which are also contextually dependent. It is therefore appropriate to examine the overall concept of localisation with a discussion of a selection of definitions to highlight both the differences and any underlying areas of commonality.

The Oxford English Dictionary (OED) [91] provides a non-specific definition in figure 3.1, this is a useful place to start as it defines, in a generalised manner, the groups of salient characteristics common to many contextual definitions by attributing location to a physical object or abstract notion.

**localisation** *n.* OED [89]

- The determination of the position of some object.
- A salient characteristic of a particular region defined by attitude or behaviour.
- Restricting something to a particular region.

Figure 3.1: Generic OED definition

The group of dictionary definitions attributable to the general concept of the process of determining the location of *some selected feature of interest* can be split into two main sub-groups.

The first sub-group is shown in figure 3.2, this contains only those definitions that

---

directly refer to methods or techniques that result in the location of a physical object that can be defined in the coordinate system appropriate to the context. The common method of gathering the data required for the localisation process in this group is the concept of locating some object using means that are indirect.

<p><b>localisation of sound</b> <i>n.</i> OED [89] The perception of the position of a sound source in space.</p> <p><b>Stereotactic localisation</b> OED [90] The accurate localisation of structures within body of living tissue by using a system that generates three dimensional measurements.</p> <p><b>Echo sounding/radio signal sounding</b> (Methodology) An active method by which the physical location of something may be determined by measuring the time taken for a sound echo or radio signal echo to return.</p> <p><b>Localisation of objects</b> (Robotics) The general determination of the physical <i>place</i> as defined by some coordinate system: where an object is actually positioned relative to a defined system viewpoint.</p>
---

Figure 3.2: Definitions that refer to the localisation of physical objects

The second sub-group is contained in figure 3.3, this sub-grouping has the common attribute that they define the location of a functional property in some generalised area of a physical organ or object. This set of definitions are notable in that they have no implied or explicit method of establishing where, or even if, a localised process will be physically located, just that the *process* itself will be located somewhere in the expected vicinity. They reflect an implicit knowledge of the context of physiological anatomy to which they uniquely refer.

The fundamental common factor applicable to both groups is that an object or functional property can be considered as being located in a certain position according to the locally defined point of reference. Although the concepts of physical and functional location are quite dissimilar the basic principle of localisation holds true for both; with the proviso that the methods used to attain the localisation data and any subsequent processing are bound by the context of the problem to be solved.

### 3. BACKGROUND

---

**Physiological localisation 1.**

The principle that specific, definable, physiological functions have relatively circumscribed locations in some particular part or organ of a living body.

**Physiological localisation *n* 2. OED [89]**

The specialization of different areas of the brain for different operations, activities, or processes.

**Localisation determination (Process)**

The determination of properties that can be reliably and repeatedly associated to a specific object, function or process.

For example; *‘The determination of molecular structures’*

Figure 3.3: Definitions that refer to the location of a functional property

The objective up to this point has been to establish a general appreciation, or understanding, of localisation by highlighting some of the areas where the term is applied. The definitions most appropriate to the work of this thesis are those of Robotics and the general concept of Echo/radio Signal Sounding.

**The process of determining the location of an object capable of mobility, at a defined point in time, relative to n-objects, in an n-dimensional coordinate system, or within the bounds defined by some mapped representation system. The resultant set of time-stamped relative positional information data can be interpreted as the route trace taken by the robot through the zone of interest.**

Figure 3.4: A Definition of localisation derived from robotics

Figure 3.4 gives the definition of robotics inspired localisation that has been devised for the work of this thesis. In effect it is an extension of the generic OED definition in figure 3.1 in that it follows some of the key attributes of the generic environment within which a generalised robotic system might be expected to operate.

This basic definition is incomplete in terms of its scope as being such a generalisation it cannot contain any additional information on how a robotic localisation system may be specified or implemented; therefore the definition of the process of localisation will always be, to a large extent, context specific.

---

The practical realisation of the latter point is very much at the core of the work of this thesis. However this definition is more than adequate to serve as the basis to derive the specification of a method with the design intention of fulfilling the requirements inherent in a 3-dimensional dynamic localisation based system.

## 3.2 Relative distance identification

In this section the objective is to introduce some of the various methods developed to determine the physical location of an object in some coordinate system. The history behind a significant number of these developments is quite often driven by the military, commercial interests, meteorological monitoring, or even occasionally, for scientific research.

The underlying mode of operation for these methods is that they can be characterised as generating a triangulated range vector that can be translated into a distance magnitude detection relative to a known physical point; specifically this is achieved without any form of physical contact between the distant object and any part of the interrogating system.

A radio or laser beam can be considered as providing positional information in a single dimension as neither elevation, directional bearing, nor the distance to any distant object that the wave energy impinges upon can be directly inferred from this simple non-vectorised beam. Most systems therefore have to generate the additional localisation information to provide the minimum set of parameters to physically locate a distant object. There have been a number of general surveys of the various methods that have been employed in various research projects in this area, specifically by (Camp et al, 2002) [16], and (Akyildiz et al, 2002) [1; 2]. In addition to these surveys there is further work that, whilst still quite general, provide more specific information to the general understanding of wireless enabled networks of the type under discussion here by (Aspnes et al, 2004) [5], (Hu & Evans, 2004) [64], (Sichitiu & Ramadurai, 2004) [112], (Balakrishnan et al, 2001) [8], and (Bulusu et al, 2002) [13].

### 3.2.1 Historical methods for the determination of relative position

Relative distance identification, or localisation, has been used to determine the relative positions of remote objects following the invention of the first useful telescopes. The telescope quickly led to the development of the first practical theodolite instrument in the middle of the sixteenth century<sup>1</sup>. The modern, accurate, theodolite was produced in 1787<sup>2</sup>. The sextant was also first invented around the same time<sup>3</sup>. This device is

---

<sup>1</sup>By Leonard Digges of Kent, England

<sup>2</sup>Jesse Ramsdens' famous *great theodolite*

<sup>3</sup>Many people contributed to the invention of the theodolite, as a result no single person can be credited with its actual invention

### 3. BACKGROUND

---

closely related to the theodolite as it is also used for determining the relative positions of known objects, in this case the latitude of a ship using the relative angle between certain visible stars; specifically, the height of the Sun over the horizon at noon and the pole star (Polaris) at night.

As the theodolite and sextant both measure two angles to a known reference point and perpendicular to each other it is entirely possible to use either instrument to do the job of the other by rotating it through 90 degrees. The fundamental difference being that the sextant measures in the vertical plane and the theodolite in the horizontal. The point being that these devices can provide a reliable and accurate method to determine the relative positions of a remote object from the local viewpoint in order to calculate the position of whichever is the unknown point. That the remote and local points are projected onto the same notional triangle infers that either point can be established, provided that the position of at least one point is known.

#### 3.2.2 An overview of contemporary range detection

There are a number of existing systems and methods to achieve practical range detection, both commercial and military, that have been developed for this primary task of generating various forms of ranging, remote object detection or relative positional information. Range detection systems that employ implicitly indirect methods to determine the distance to a remote object are also likely to be subject to some level of non-indeterminacy in their accuracy, reliability or repeatability of the generated ranging and relative positional data.

**These systems can be divided into two broad groupings.**

- Time elapsed systems
- Time difference of arrival systems

These system groupings both share a common characteristic in that they are based on the speed of transmission of electro-magnetic waves (radio, light or sound) through one or more media, (the atmosphere, the vacuum of space, or both) to derive a distance magnitude between the transmitter and the distant object to be detected. Where they differ is in how they use this magnitude information to establish the required physical location. On its own a simple magnitude cannot provide a definitive position, to be useful this basic information must be represented relative to a defined point in a known reference frame which itself defines a minimum of a polar or Cartesian coordinate geometric structure. The distance magnitude, in the form of a Euclidean vector, can then be further translated into a set of defined coordinates relative to a suitable frame of reference. The reference data is commonly provided by an externally generated post-processing system or method appropriate to the specific context of that system.

---

For systems that rely on the constant speed of electro-magnetic waves, the assumption is that the speed of radio waves and the speed of light are identical. In a vacuum the speeds are almost the same, in the case of propagation through air there is a small, but quantifiable, difference in absolute speed; for the systems being considered here that operate over relatively short distances the effect of this will be negligible, and this assumption can be deemed valid.

One notable exception to this assumption is the Global Positioning System (GPS) as propagation through air and the near-vacuum of extra-terrestrial space will both be encountered. In this case the distances through air and vacuum *are* sufficiently large to cause a noticeable change in time taken to travel the same distance through air and the near-vacuum. Also to be considered in terms of positional accuracy, (mainly by the designers of GPS equipment), are the effects of ‘*Special*’ and ‘*General*’ relativity as discovered by (Einstein et al, 1920) [36], these result in time differences between the clocks on the satellites and those on Earth of seven and forty-five microseconds per day for ‘*Special*’ and ‘*General*’ relativity respectively which must be compensated for if positional accuracy is to be obtained.

### 3.2.2.1 Time elapsed systems

Systems that determine distance, or range, using the elapsed time between events have a shared fundamental mode of operation. The distance between two objects can be determined through a simple calculation based on the speed of electro-magnetic or sound waves and the length of time taken for that *signal* to propagate the space between the transmitter, be reflected by the target object and to travel back to the receiver. The actual time taken, associated speed and subsequent magnitude of the distance will depend on the type of wave and the constitution of the medium through which the waves propagate; the calculations are otherwise similar.

*These systems can be divided into two groupings*

- Sound based ranging
  - **SO**und **N**avigation **A**nd **R**anging (**SONAR**)
- Light based ranging
  - **LI**ght **D**etection **A**nd **R**anging (**LIDAR**)
  - Cloud base recorders
    - \* Laser Ceilometer
    - \* Modulated visible light based

These systems determine a direct *line-of-sight* distance magnitude calculated using the speed of the particular type of propagated waves through the appropriate medium

### 3. BACKGROUND

---

and the amount of time taken by those waves for the return journey between transmitter and target object, as covered in the work on acoustic ranging in resource-constrained sensor networks by (Sallai et al, 2004) [109]. In terms of localisation the data produced in this manner will always be limited to indicating the simple magnitude of the distance between the objects. Where full 3-dimensional positional information is a requirement then this must be supplied through additional data generated by the transmitting device itself or through external sensors giving a precise or more general direction. In the case of SONAR a general direction will be sufficient to locate an object, others, such as RADAR or light-wave based devices may demand, and be capable of providing a more accurate relative position in 3-dimensions.

#### 3.2.2.2 Time difference of arrival systems

The fundamental mode of operation for this type of system is the calculation of the difference in time taken for a known radio wave transmission to reach two (or preferably many more) receiving stations as described by (Gustafsson & Gunnarsson, 2003) [53]. The time difference between these receiving stations will therefore be related directly to the geometry that describes these relative positions of the receiving stations. Other localisation systems based on the intersection of radio beams include GEE (the name derived from the hyperbolic Grid described by its operation), the OBOE system by (Cunningham et al, 1984) [23] and the Long Range Navigation(LoRaN) system described by (Davis, 1945) [26] and its many commercial variants.

#### 3.2.2.3 GEE, LoRaN and OBOE

The GEE and LoRaN systems are all closely related, but differ greatly in how they are implemented, this is mainly in how the positional information is calculated, but the fundamental mode of operation is essentially the same; this section uses the OBOE system<sup>1</sup> as the example. This group of localisation systems employ two independent timed radar signals that can be interpreted through various means to provide a fixed geographical position in real-time. This is achieved either by a ground station tracking the course of an aeroplane and sending course correction data, or by the navigator on the aeroplane plotting the information independently. The most operationally effective of these systems was the OBOE system that used radio transponder technology to send two signals to an aeroplane flying in the vicinity of the planned location. Each signal (on a different frequency) is received by the appropriate transponder on the aeroplane and then re-transmitted back to the originating ground-based transmitter; the round-trip time for each of the signals is calculated and converted to a distance magnitude between the current position of the aeroplane and the known positions of the ground-based transmitters.

---

<sup>1</sup>OBOE is not an acronym, the word refers to the sound made by the radio transmissions being reminiscent of the woodwind instrument when replayed through a loudspeaker, although an acronym was created retrospectively.

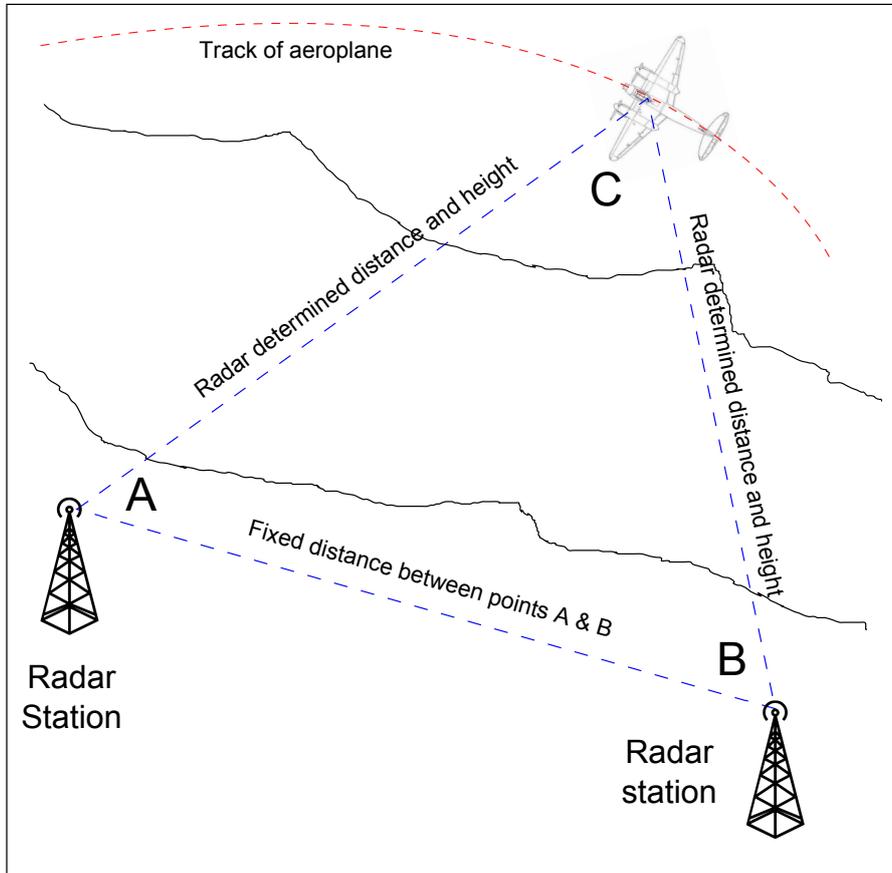


Figure 3.5: Radar based triangulation

The basic method of determining the position is through tri-angulation for simple *'line-of-sight'* calculations, or tri-lateration if the distances are significant where the curvature of the planet will become a factor. In order to calculate a fixed position for the aeroplane the two transmissions must be sent from geographically different points at known latitude and longitude points thereby giving a known distance between them, (points A & B in figure 3.5), effectively these two points form one side of a triangle. The third point of this notional triangle, (point C in figure 3.5), is then calculated using the radar determined lengths of sides A:C & B:C; where they *cross* gives the current position of the aeroplane. The use of RADAR in this example provides height information in addition to the bearing and distance data in the process of *'creating'* the notional triangle.

### 3.2.2.4 The Global positioning system

The Global Positioning System (GPS) satellite based technology now in widespread commercial and military use provides real-time positional, time, bearing and velocity data, A major difference between GPS and the OBOE type is that the positional data

### 3. BACKGROUND

---

is only available at the receiver; due mainly to the GPS generally being a receiver based system and not being equipped with the capability to (re-)transmit a received transmission, other considerations are power availability and the non-directional antennae commonly fitted to these devices. At the most basic consideration, the GPS system functions using the same underlying time/distance calculations as the OBOE system.

There has been much research carried out into the utilisation of geo-spatial information in wireless enabled sensor network research by (Bulusu et al, 2000) [14], (Heidemann & Bulusu, 2001) [56], and (Romer, 2003) [105]. Specifically these require the use of GPS modules as part of the networked elements to provide the 3-dimensional location information; this remains an option for the work of this thesis, but there are considerable operational drawbacks that would require careful consideration. GPS can determine a physical position with considerably more accuracy and precision than any other system primarily because of the amount of temporal data available for that purpose.

The main difference is the amount of post-processing required to be performed by the GPS receiver. The raw data used for calculating a position is all sourced from signals received by a number of orbiting GPS satellites, at the time of publication there are 24 in orbit with approximately 8-10 above the horizon at any one time and therefore *visible* to the GPS receiver. The ultimate accuracy depends on the *'level of service'* made available by the owners of the satellite system, this is reflected in the number of satellites made available for each positional calculation and the resolution of the data transmitted to the GPS receivers; although this aspect has seen significant recent improvement in recent years the original GPS<sup>1</sup> remains a military system and the level of service could be reduced, or completely removed, at a future date in response to world events probably with little regard to any commercial contracts. The GPS receiver calculates its position by the comparison of precisely timed signals from the GPS satellite messages that are constructed from the following:

- The (very) precise time of message transmission (time synchronisation of all satellites is controlled by an Earth station)
- Precise orbital information contained in the ephemeris (a generic term for the table of values containing the current positions of astronomical objects - including satellites).
- The system health and approximate orbits of all GPS satellites (commonly called the almanac).

The GPS receiver determines the transit time of an individual message by comparison of the time information contained in the message transmitted by a satellite and the *local* time of the receiver itself. This process is repeated for all visible satellites for each epoch resulting in a set of distance magnitudes between the GPS receiver and

---

<sup>1</sup>A European conglomerate has recently installed the ostensibly commercial Galileo satellite system to alleviate this concern: [37]

---

each discrete satellite. This is the point at which the GPS differs most significantly from OBOE type systems as GPS operates in 3-dimensions, over far greater distances, and with the light speed variations and general/special relativity issues as mentioned in the introduction to this section. The set of distance magnitudes are converted to a physical 3-dimensional position (in terms of latitude, longitude and altitude) using the physical location of the satellites contained in the ephemeris as the origin of a set of spheres with the resultant (calculated) intersection indicating the position.

### 3.3 Radio Signal Strength Indication

The IEEE 802.11 descriptive document (Crow et al, 1997) [22] defines the Wireless Local Area Network (WLAN) protocol applicable here, this is the same communication system commonly found in commercial and domestic router wireless internet environments designated as 802.11(b|g|n). Other wireless protocols also specify Radio Signal Strength Indication, or another similar metric for the purposes of signal strength or quality; however the 802.11 protocol is specific to the variant of the Crossbow™ Smart Dust wireless system originally selected as part of the model abstraction.

The Receive Signal Strength Indicator (RSSI) metric is also defined in the IEEE 802.11 document as the relative signal strength in a wireless environment as a simple indication of the power level being received by the antenna. In general a higher RSSI value indicates a stronger received signal level. Therefore the RSSI metric represents the transmission power density as *perceived* by a wireless receiver after translation by the internal electronic circuitry of the receiver, commonly employing a logarithmic conversion function specific to the receiver hardware implementation.

RSSI measurements are restricted to the range ‘0 to 255’ as this can be represented in a single byte as an unsigned integer, the actual maximum value of **RSSI\_Max** will be dependent on particular hardware but is usually in the range of ‘0 to 100’. The 802.11 standard does ‘not’ define any relationship between RSSI and power level, (in milliWatts (mW) or deciBels referenced to one mW (dBm)). The resultant RSSI data therefore equates the signal strength as a range of values described by the specific logarithmic relationship relative to some notional zero point. RSSI and its application to localisation has been the subject of a number of research papers, one of the papers that proved most useful to the work of the thesis is that by (Hsieh et al, 2004) [63] where they considered the generation of maps describing the general relative position of a set of *robots*. This paper relied on a centralised data-base to place various obstacles and robots in the environment. Whilst this approach was successful it is not appropriate for the work of the thesis as the environment that contains a number of free-flying element is likely to be highly dynamic. It is likely there will be insufficient time to reliably calculate the positional information; although it can also be argued that the input data-set used here does, with some justification also represent a data-base of positional information, albeit one dynamically generated and inherently relative in structure.

## 3. BACKGROUND

---

### 3.3.1 Idealised electro-magnetic wave propagation

In the idealised environment that is devoid of all sources of external adverse interference, wireless signal propagation the received power density can be represented as a constant conversion function that follows a simple repeatable relationship of attenuation with distance following a defined logarithmic law. The propagation of electro-magnetic waves (radio, light, X-rays, etc.) through free space is described by the inverse-square law,  $P_{Density} = \frac{1}{r^2}$  (where  $r$  is the magnitude of the distance between source and remote points).

The power density of an electro-magnetic wave propagated through idealised free-space will be proportional to the inverse of the square of the distance from the source.

Figure 3.6: The propagation inverse-square law

The idealised inverse square law is defined in figure 3.6, thus for every doubling of the transmission propagation path distance the power density of the radiated wave as received will be reduced by a factor of four. The power density of the notional beam area at the reception point is proportional to the product of the electric and magnetic field strengths; both of these aspects being halved thereby resulting in the factor of four reduction.

### 3.3.2 Radio Signal Strength Indication in the real-world environment

The presence and constituents of the terrestrial atmosphere, that defines the real-world environment being considered here, has an undeniable and strong influence on all radio frequency electro-magnetic waves propagated within it. The level of influence, commonly, but not exclusively, will be one of attenuation, the level depends on a number of parameters; the most relevant are the frequency of the wave, the transmission medium through, and the overall distance of transmission.

#### Section headings of some of the more common external influences

- The atmosphere and surrounding obstructions
- Temporal Transmission Collisions
- Receiver Sensitivity and Transmitter Power Variations
- Signal Masking and Antenna Orientation Variations
- Relative antenna orientation

---

All of these influences can cause interference are likely to have a deleterious effect on the quality of the wireless link and therefore on the dependability of the RSSI metric when being used as the means to determine distance between transmitter and receiver. Therefore the idealised case for signal attenuation given in section 3.3.1 will only be valid in an absolutely idealised *free-space* environment. As a result of this transmission power densities are more appropriately represented by an estimation based on an inverse-cube law,  $P_{Density} \approx \frac{1}{r^3}$  (where  $r$  is the magnitude of the distance between source and remote points). This reflects the effect some external interference can have on the relationship between radiated power and distance. These aspects are explored through empirical analyses of radio signal strength variability and proximity quantisation as described in (Lymberopoulos et al, 2006) [82] and (Patwari & Hero, 2003) [94] respectively.

The external influences affecting radio transmissions are significant, numerous and, above all random; as a result neither of the transmission power density shown above, and in section 3.3.1, represent actual reality; they cannot be relied on to give a reliable and repeatable indication of the distance between a transmitter and receiver pairing for any given transmitted power density. As a direct result the external effects on the overall quality of RSSI can result in one of the main causes of indeterminacy present in the problem search space, thus contributing to the concept of this being a Non-deterministic Polynomial Hard (NP-Hard) problem.

### 3.3.2.1 The Atmosphere and Surrounding Obstructions

The real-world transmission medium is therefore far from being an idealised environment, it is beset by many atmospheric variables. This interference can cause significant degradation to the quality of the wireless link; a likely consequence of this interference will be associated data stream corruption.

**These sources of interference are many and varied:**

- The prevailing meteorological conditions
  - The moisture content of the air
  - The presence of electrical activity (lightning)
- Multi-path reflections from obstructions
  - Multi-path fading and shadowing in the RF channel
  - Reflections from aircraft, birds, buildings and the ground itself
  - Temperature inversions (with increasing altitude)

## 3. BACKGROUND

---

### 3.3.2.2 Temporal transmission collisions

The likelihood of transmission collisions affecting the reliability and overall quality of wireless connections can be viewed as being directly proportional to the ever increasing quantity of such devices and the diverse and proliferating range of new uses being found for them. The probability of interference from another nearby transmitter on a similar or harmonic multiple frequency to the primary frequency is not therefore likely to reduce. Even if the operating frequency of an unwanted transmission is not the same there is still the possibility of harmonic interference causing unwanted signals being passed through the intermediate frequency discriminator stages of the receiver with the resultant alteration of the apparent received signal strength; the effect of the latter can cause attenuation or cause artificial augmentation. Unfortunately there is little that can be done to alleviate this source of interference unless all of the transmitters are under synchronised control.

### 3.3.2.3 Receiver sensitivity and transmitter power variations

Mass produced electronic systems are, on the whole, produced to a very high level of quality and reliability, but there will be variations between such devices even when made consecutively. In the effectively deterministic world of the digital system, where apart from rare occasions where corruption has caused the signal voltage levels signifying the ‘*zero/off*’ or ‘*one/on*’ states there is can be no variation between them. Wireless transmissions, (in this application), are analogue and do not have this level of certainty between what was actually transmitted, in terms of radiated power and the information contained therein, and what is actually received; even assuming a notionally ideal transmitter/receiver and antennae.

The receiver end of the system has an equally problematic environment in which to operate as the received signal is likely to be tiny fraction of the transmitted power, and has potentially been subject to all manner of interference. In addition the variation between apparently identical transmitters and receivers adds a further level of uncertainty into the quality of the distance/received field strength relationship via the wireless link.

### 3.3.2.4 Signal masking and antenna orientation variations

The majority of wireless transmissions are *line-of-sight*, especially those that employ the Frequency Modulation (FM) system of encoding the data onto the underlying, effective, carrier wave. Signal reflections are still possible, as has been stated above, however an obstacle could still have the effect of blocking the transmission thereby making it invisible to a potential receiver, or at the very least of attenuating signal to such an extent that, in terms of the principle under discussion here, the signal strength as received would bear little resemblance to that transmitted and therefore likely to be of limited use.

External influences that are significantly likely to cause significant variations to the distance/received field strength relationship are the effects introduced by the type and

---

configuration of the antenna systems employed by the transmitter and receiver. Both transmitter and receiver antennae will exhibit non-linearity with respect to the sensitivity, (in the case of the receiver), and the power density, (in the case of the transmitter), that will cause variations in the signal strength depending on where in the radiation pattern the signal reception occurs.

The radiation pattern of the transmitter antenna is the most significant in its effects, and this becomes progressively less so with increasing distance. The effective radiation reception pattern of the receiving antenna will also have an effect on the distance/received field strength relationship but is significantly less than that of the transmitter.

Once the antenna radiation pattern characteristics are known then it may be possible to mitigate the effects by various compensatory methods.

### **3.3.2.5 Relative antenna orientation**

The relative orientation of the antennae in a transmitter/receiver pairing will also directly influence the strength of the received signal, in effect this can be viewed in a similar manner as that of the radiation pattern from the previous section. For maximum signal strength the orientation needs to be perfectly aligned, any variation, like the radiation pattern will, (usually), result in an attenuated signal.

In most installations the orientation relationship can be controlled, thus mitigating the effects to a known quantity. Where either, or both, of the transmitter and receiver elements are not fixed in position or orientation then this aspect is likely to be responsible for considerable variation in the distance/received field strength relationship.

### **3.3.3 Radio Signal Strength Indication as an input data-set**

The inter-element signal strength distance signal strength data is used as the primary source of data by which the 3-dimensional positional data for each Smart Dust Sonde device can be derived. The RSSI data is a simple logarithmic magnitude metric representing the transmission power as '*perceived*' by a wireless receiver after translation by the internal electronic circuitry of the receiver. This magnitude metric can be translated into a '*reasonable*' representation of the distance between the transmitter and receiving elements at a particular moment in time. There are many external influences on the reliability of this data that ensure that it should always be considered as being non-deterministic, these influences can cause largely unpredictable variations in the accuracy and repeatability of any instantaneous RSSI value representing a distance magnitude.

In addition to being non-deterministic the individual distance metrics, when considered in isolation, are limited still further by the non-vectored magnitude aspect which does not have any other associated positional information. They become much more useful when the magnitude metrics from all Smart Dust Sonde devices are considered as a com-

### 3. BACKGROUND

plete data-set; in this state the spatio-temporal 3-dimensional geometric data encoded in the data-set can be ‘*discovered*’ by a considered algorithmic approach designed to rebuild any such ‘*hidden*’ geometry. The supposition is therefore, that encoded within this data-set, is an indirect spatio-temporal representation of the 3-dimensional geometry of the discrete Smart Dust Sonde devices in the cluster at the specific moment in time when the set of transmission and reception events occurred.

Field	Description
Type	Text packet type Identifier; "NULLPKT" "SYNCPKT" "RSSIPKT"
Ident	Numeric value of packet type Identifier; (See definitions below)
Source	Unique ident from SENDING element (the sender of this data packet)
Remote	Unique ident from REMOTE element (data packet receiver, adds RSSI/SYNC)
Total	Total number of packets for this (originating) element
Time	Globally set time epoch identifier
Payload	RSSI value in text form or GW_BROADCAST/RR_BROADCAST
End	End of packet marker - Currently set to ‘;’

Definitions:		
Identifier	Value	Description
NULLPKT	0L	Missing data
SYNCPKT	1L	Gateway packet to cause all element synchronisation
RSSIPKT	2L	Packet containing RSSI data
GW_BROADCAST	100L	RSSI data originates from the gateway
RR_BROADCAST	101L	RSSI data originates from a remote element

Figure 3.7: Synchronised inter-element packet format description

Field identifiers								
Field	Type	Ident	Source	Remote	Total	Time	Payload	End
BYTES	12	4	4	4	4	4	30	4
TYPE	Byte array	Enumerated	Integer	Integer	Integer	Double	Byte array	Byte array

Figure 3.8: Synchronised inter-element packet format definition

The input data-set files are essentially a set of *packets* of data with a format intentionally similar, (in principle), to that used as the main data transport structure of a normal computer network. The description of the fields and the data (byte) format as shown

---

in figures 3.7 and 3.8 respectively. An example of the file structure is given in appendix D.

### 3.4 A generic Radio Sonde monitoring system

The Radio Sonde [123] devices and associated receiving systems are, in effect, miniature weather stations designed to monitor upper-air spaces. They are *flown* aloft by helium filled weather balloon or through being dropped from an aircraft in the case of the Drop Sonde. It is the latter type that is the more precise model for the motivation of the work of this thesis, although both are functionally similar so either would suffice.

Commercial Radio Sonde systems have many shared attributes and their fundamental operation may be simplified to that of a system of defined function modules. In meteorology the Sonde system still provides a vital part of the data for weather forecasting and climatology. Modern systems, such as satellites, can provide an excellent source of data through temperature and ozone profiling, infra-red and visible light imaging, and cloud-top monitoring. However these techniques, at the time of writing, are not yet capable of providing the same depth of detailed observation provided by a Sonde due to it *monitoring directly in the locality*. It is reasonable to conclude that the Sonde will have a significant role to play for the foreseeable future.

#### 3.4.1 Sonde monitoring and wireless communication

The generic Radio Sonde communication and data processing system consists of one or more remote airborne transmitter modules, a radio receiving station capable of multiple channel discrimination and a data storage and post-processing system appropriate to the actual monitoring task.

The wireless communication between the Radio Sonde and receiving station is, usually, via a simplex wireless link operating on a range of reserved frequencies adjacent to 403 MHz or 1680 MHz. The transmitter output power of these devices is between 100-300 mW which results in sufficient range in free air to enable the use of a non-directional antenna system, thus obviating the need for a complex tracking system to ensure that wireless link is maintained. However for validation of the monitored meteorological parameters and the extrapolation of other data types there is still a requirement for a system to track the 3-dimensional position of the Sonde itself with respect to time.

#### 3.4.2 Meteorological Radio Sonde synoptic data types

A typical meteorological Sonde generates a synoptic data report that consists, primarily of a set of directly monitored parameters, and others that can be inferred by calculation or through the tracking system. In addition to the determination of the precise 3-dimensional location where the data was measured the tracking system also supplies the timing information to assist in the reconstruction of the data stream.

### 3. BACKGROUND

---

#### Directly monitored parameters

- Barometric air pressure
- Air temperature (and dewpoint)
- Relative humidity
- Cosmic ray readings at high altitude

#### Additional parameters inferred through positional tracking

- Wind (speed and direction)
- Altitude
- Latitude & Longitude

The sensor readings (or soundings to use the original terminology) are taken at a fixed rate (usually around 2Hz) and this relates to the immediate area surrounding the track of the Sonde on its ascent or descent; this synopsis is then periodically transmitted to a receiving station using a simplex wireless link.

Ultimately this data is plotted onto a tephigram [121] for analysis purposes; this is a graphical thermodynamic diagram representation of the pressure, temperature and humidity upper-air observations made during the vertical sounding of the atmosphere. The tephigram has three logarithmic axes configured in such a way as to correct for certain changes in atmospheric conditions during the flight. Expert interpretation of a tephigram can reveal significant prevailing meteorological conditions that are powerful aids to weather forecasting.

#### 3.4.3 The types of Radio Sonde devices

The airborne element of the Radio Sonde family of devices are effectively self-contained weather stations optimised for upper-air environmental monitoring tasks. These are complete systems with enclosures, sensor suites, software, operational modes and deployment methods purposely designed for an intended task.

The original Radio Sonde was the *Kew* Sonde designed by (Dymond, 1947) [34] for the UK Meteorological Office in 1939, the Mk2. followed in 1945 and remained in use until the early 1960s. These devices were designed specifically for ascending upper-air weather observation in the region between ground level and the Troposphere, between about 65000 and 115000 feet.

The current commercially available sonde devices are functionally similar to the original

---

design although they are now made using modern sensors and electronics. The generic types of Sonde device of interest here are the *standard* Radio Sonde, the Drop Sonde and its technically similar Drift Sonde derivative.

The fundamental differences between the types of Sonde can be summarised thus:

- The mode of deployment.
- The deployment dependent packaging of the instrument.
- The direction of travel during the data monitoring *flight*.

### 3.4.4 The Radio Sonde

The standard Radio Sondes [123] are constructed using lightweight materials to enable them to be deployed by being *flown* aloft by a relatively small helium or hydrogen filled weather balloon as shown below in figure 3.9. When the balloon reaches the desired maximum altitude (controlled to a rough degree by the balloon gas pressure at launch), the surrounding atmospheric air pressure is low enough to cause the balloon to burst, the data generated from the hurtling descent that follows is not generally monitored.

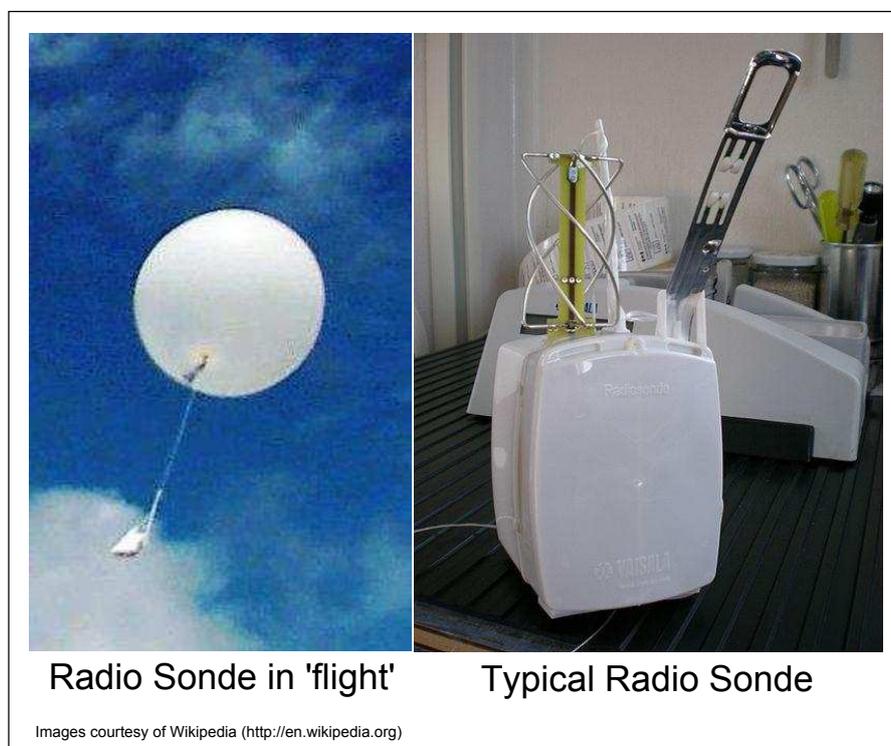


Figure 3.9: A generic Radio Sonde

### 3. BACKGROUND

---

The Sonde must be tracked during the active data monitoring phase of the flight, the purpose of this is not to enable retrieval as Radio Sondes are generally regarded as consumables, but to provide the necessary 3-dimensional positional validation of the monitored data by the recording of time and location information. There are several methods used for tracking the Sonde, the most common method is by tracking with a *Wind-finding* RADAR (Jaatinen & Elms, 1999) [67], depending on the resolution of the Radar in use this can be very accurate. The positional data that is generated is usually in the form of a time-stamp, range, compass bearing and elevation (also in degrees), this can be referenced back to the actual geographical location of the Radar for conversion to latitude and longitude. The positional data is used in the extrapolation of wind speed and direction directly from flight track of the Radio Sonde as this is directly influenced by these parameters. It is also possible to fit the Radio Sonde with GPS modules, this is not standard practise for the UK Meteorological Office [120] however.

#### 3.4.5 The Drop Sonde

The Drop Sondes [124], an example of which is shown in figure 3.10 provides essentially the same data stream as the ‘*standard*’ Radio Sonde, but due to the mode of deployment is a substantially more robust device than the standard Sonde. The Drop Sonde, as the name implies, is deployed by being dropped from an aeroplane directly into an particular area of interest, such as a storm cloud.



Figure 3.10: The Vaisala RD94™ Radio Sonde

The rate of descent of the Drop Sonde is limited to around 20 knots by a parachute assembly which also houses the antenna. After deployment of the Drop Sonde the aeroplane then engages in a suitable flight pattern to maintain wireless communications with the Sonde on its descent through the air-space. Although the rate of descent is controlled by the parachute the Drop Sonde will still have considerable momentum when it reaches the surface, thus it is usually deployed over open sea for obvious reasons of safety.

Whilst the standard Radio Sonde can be tracked quite easily, the method and general location of Drop Sonde deployment preclude the use of Radar for tracking. If accurate localisation data is required then Drop Sondes can be fitted with GPS modules, these are especially useful if detailed wind speed and direction profiles are required as part of the research data stream. There are some technical difficulties with using GPS, these and the cost are examined in a later section.

### 3.4.6 The Drift Sonde

The University Corporation for Atmospheric Research (UCAR) Drift Sonde [118] is part of a research system with the primary purpose of providing data over a period of several weeks to aid research into certain meteorological phenomena. This has been included here as some aspects of its operation have interesting parallels with the work of the thesis.

In operation the Drift Sonde is a combination of the Radio and Drop Sondes, in this system a balloon resembling a larger version of that for a Radio Sonde ascends to the lower stratosphere (around 65000 feet) where it is able to Drift with the prevailing winds, hence the name of the system.

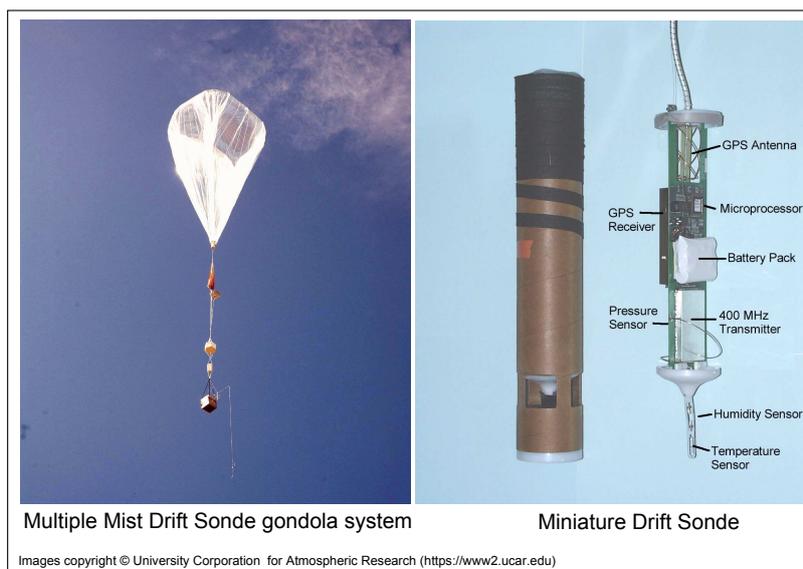


Figure 3.11: The UCAR Drift Sonde Deployment System

### 3. BACKGROUND

---

#### System Capabilities:

- GPS enabled Sondes
- Flight duration of a few days (zero pressure balloon) to several weeks (super pressure balloon)
- Balloon flight altitudes 100hPa to 30hPa
- Payload up to 50 miniature Drop Sonde devices
- Web access interface

The deployment system balloon is shown in figure 3.11, this supports a *gondola* that contains the communication and data storage system and a number of small and light development Drop Sondes. These are released as discrete devices by remote control. The Drop Sonde data is stored by the gondola system until accessed by the remote system. This system has been in operational use for a number of years and has contributed significantly to **THE Observing System Research and Predictability EXperiment** (THORPEX) [115] and the **African Monsoon Multidisciplinary Analysis** (AMMA) by (Redelsperger et al, 2006) [99].

#### 3.4.7 Some general Radio Sonde system limitations

A factor common to all Sonde types is that the operation of the simplex wireless system limits communications to one-way, in this instance transmitted by the Sonde to the receiving station. The consequence of this is that they are isolated from communicating with any other simultaneously deployed Sondes, in general however as they are usually deployed singly this is not usually a particular drawback. More critically perhaps, is that it does not allow for any additional data types to be generated by any potential interaction between Sonde devices, the difficulties of tracking discrete Sondes notwithstanding. As a result the meteorological synopsis the individual Sondes generate is defined solely by their discrete and isolated track through the upper-air region.

A substantial part of the abstraction underlying this thesis is the concept of a multi-element co-operating clustered Sonde that could be deployed in a similar manner as for the Drift Sonde or by aeroplane deployment. Although the Drift Sonde system has the potential to release a number of Sondes simultaneously as a cluster, at the time of writing, published information has not been found stating that this has been implemented. In any case for the Sonde elements to properly co-operate imposes a requirement for additional methods of localisation for all elements in the cluster; this forms the novel software and optimisation algorithm modelling aspects of the research in this thesis which will be explored in a subsequent chapter.

---

### 3.5 An Overview of evolutionary algorithms

Evolutionary Algorithms (EA) are a subset of evolutionary computation in the general field of Artificial Intelligence (AI), the objective here is intended to introduce the concept at the highest operational level, details of the EA implementation in run-time the software modelling system code used throughout the thesis is described in the sections relevant to the specific aspect in chapter 4. At its simplest interpretation an EA uses mechanisms inspired by evolutionary theory by (Darwin, 1859) [25] to generate new genetic material, including: ‘*survival-of-the-fittest*’, reproduction, genetic mutation, and child chromosome generation through parental recombination. In effect EA mimic some of the more fundamental aspects of ‘*natural selection*’ to repeatedly create, or evolve, new generations of child chromosomes.

Whilst this process is frequently termed ‘*optimisation*’, an EA is intrinsically random or stochastic in operation; therefore it does not, and cannot, guarantee that any result will ever be absolutely optimal. As EA are essentially non-deterministic the very real possibility exists that no algorithm will be capable of finding an optimal solution, or of determining if one actually exists; this is particularly so as the non-determinacy also ensures that as far as the author is aware, there are currently no techniques to reliably determine if a candidate solution is actually optimal; this is a strong reason why EA are used in the first instance. This is summarised in figure 3.12.

*An EA is a set of instructions that can be iteratively applied to a particular problem, or search space, in order to be able to determine a candidate solution to a specific problem that may be otherwise intractable in terms of processing time due to the size or complexity of the problem search space.*

Figure 3.12: Simple definition of an evolutionary algorithm

The candidate solutions generated through the iterative evolutionary process are in the form of individual chromosomes in a (generally) closed population. An objective ‘*fitness function*’ designed to determine the suitability an organism, (that is the chromosome problem space representation) to stay in the evolutionary process; specifically whether it will continue to thrive, merely survive, or die out completely in the context of the problem space environment in which they ‘*live*’.

The fitness function, along with the stochastic element are the primary determinants that ‘*direct*’ the generation of new genetic material during the evolutionary process by the application of the algorithm logic through the mutation and crossover operators, the result is assessed for fitness at each iterative stage. EA are essentially stochastic,

### 3. BACKGROUND

---

or random, based algorithms, excessive levels of evolutionary pressure will frequently disrupt the generation of fit candidate solutions with the result that the overall fitness of the population will be reduced; it is important therefore to ensure the problem space representation and algorithm do not impose too strong an influence on the evolutionary process.

#### 3.5.1 A simple evolutionary algorithm

In general, EA have quite a simple internal algorithmic structure when reduced to the essential and defining characteristics. The ability to generate candidate solutions to some of the more intransigent problem spaces is, in part, due to the ‘*investigative power*’ imbued by the iterative processing implicit in the basic algorithm. Essentially the algorithm repeatedly tries new candidate solutions that have been ‘*slightly modified*’ until it finds one that is ‘*better*’ (according to the objective fitness function), than the solution currently considered to be best; put another way, without the power of modern computers it would be difficult to harness the strong evolutionary capabilities of EA.

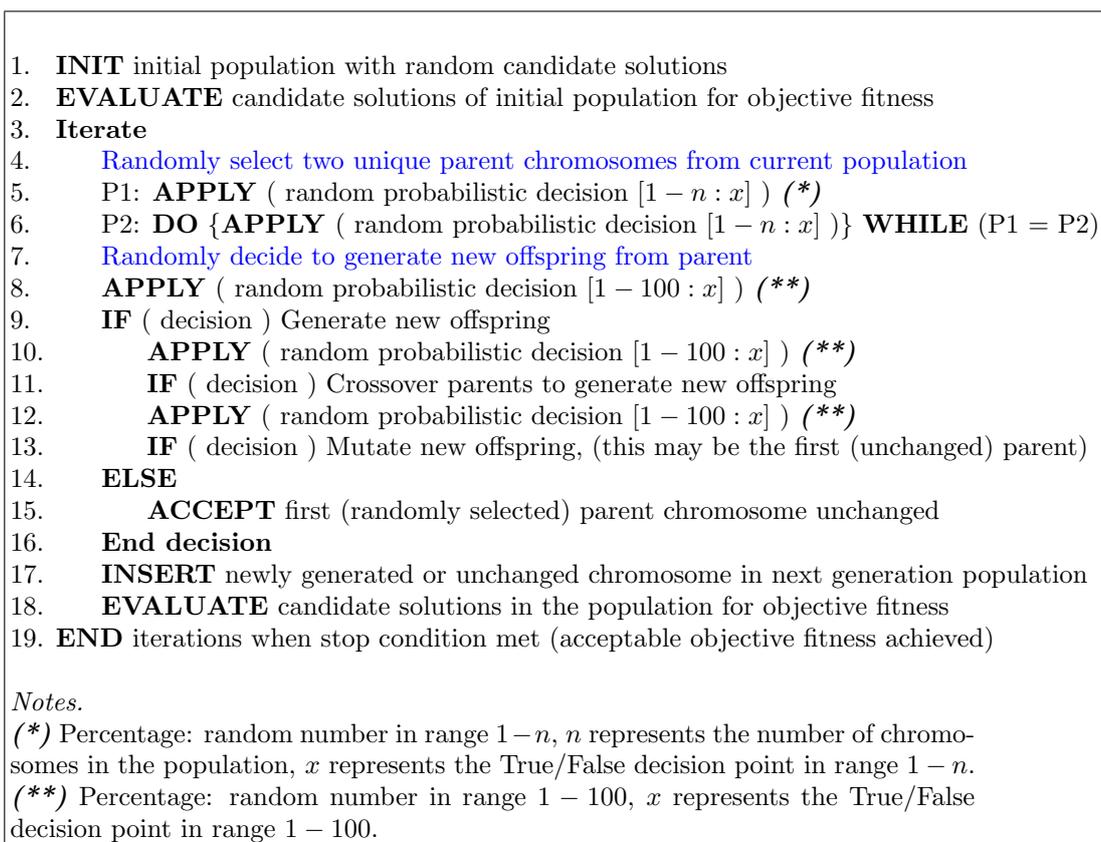


Figure 3.13: Simple evolutionary algorithm pseudo-code

---

A simple evolutionary algorithm that demonstrates the fundamental concept is shown in figure 3.13, this pseudo-code illustrates the iterative structure appropriate as the basis for the most common EA problem space implementations. The first stage (lines 1 & 2) involves the simple ‘*initialisation*’ of the chromosome population with a set of, (usually), randomly generated initial candidate solutions. The second part of this stage is the evaluation of the initial objective fitness values; generally this event happens only once during the run-time of an EA, unless a catastrophic reset is required.

The next stage of the process is ‘*Selection*’, (lines 4 to 6), two unique parents are required to provide the base genetic material from which new offspring may be generated; both parents are usually selected through some form of competitive selection process, for example, Tournament or Roulette selection. The actual method chosen for this may be problem space dependent to some extent and can only be reliably determined through knowledge of the problem space characteristics or empirical means as to which will be most appropriate; common competitive selection methods are Tournament/Roulette/Accept least fit, although there are a number of others.

At this point a decision is required regarding whether to allow the first selected parent to be transferred directly to the next generation population completely unchanged, or to produce a new offspring, (lines 8 to 16). This stage is controlled by a (user set) parameter that generates a percentage probability value as the basis of the random decision process, in this example, if the decision is found to be false the parent is accepted unchanged (line 15), otherwise it is used to generate new offspring, (lines 10 to 13)<sup>1</sup>.

The creation of the new offspring generally has two phases that are frequently, though not always, subject to a percentage probability decision as described previously; it should be noted that the pseudo-code as shown makes it possible that the crossover, (and mutation), lines may be skipped in any iteration if both random control variables are false. The first of these phases, (line 11), is to generate the new offspring by combining genetic material from both parents into a new offspring, (ie. a new candidate solution), this process of ‘*Recombination*’ uses existing genetic material that already exists, no new material is created.

The next phase is ‘*Mutation*’, in this part of the process some level of mutation appropriate to the format of the chromosome and problem space implementation is applied to the newly created offspring, (or unchanged first parent if the crossover did not take place), from the previous (recombination) phase, (lines 12 & 13). In this example the percentage probability decision is applied to decide if this should take place or just allow the new offspring to remain as created by the recombination phase.

Finally the new offspring, (or unchanged first selected parent), are placed in the next

---

<sup>1</sup>It is quite feasible to generate a pair of offspring at this point, these can also be subjected to competitive selection, however this approach is not depicted in this pseudo-code.

### 3. BACKGROUND

---

generation of population candidate solutions and subsequently evaluated for objective fitness, (line 18). If the stop condition is met, this can be a set number of iterations or a pre-set fitness value, the iterative process is terminated and the final ‘*winning*’ candidate solution and associated objective fitness value are presented as the results for this run of the EA on the problem space in question.

As stated previously the EA is fundamentally stochastic in its underlying mode of operation and, as such, there can be no guarantee that an optimal solution will be found. To successfully employ an EA in a computational environment the salient aspects of the problem space must first be defined, simplified if necessary, and subsequently encoded in a suitable set of software data structures, algorithms, semantics, and access methods. The rules of any heuristics encoded in the active optimisation algorithm are then applied iteratively until the ideal, or best possible, candidate solution is produced.

#### 3.6 An overview of the object oriented paradigm

The main purpose of the Object Oriented Design (OOD) paradigm as described by (Henderson, 1993) [57], amongst others, is to provide the intermediate specification and design stage to support the process of mapping real-world problems to the computational domain. OOD comprises a set of principles that, individually or as a sub-set, can encompass specific aspect(s) defining salient aspects of a generic real-world problem specification.

1. ***The class/object definition***, this is the elementary functional entity in an OOD based software design, it defines the design blueprint of objects that can be instantiated as run-time instance(s). This is a defined container that provides access control through a function/method interface to the raw data and any associated meta-data attributes, fields, or properties specific to a problem definition.
2. ***Data encapsulation***, a fundamental principle of OOD, the protection of defined data groupings within the boundaries of a discrete object instance; data can only be accessed through a correctly designed function/method interface.
3. ***Inheritance***, this principle defines the potential facility that allows sub-classes, (and therefore object instantiations), to inherit primary function/method implementations, interface definitions and data from a parent superclass; exactly what is accessible by child sub-class(es) depends on the hierarchy of specified access control modifiers. Inheritance through parentage can extend back many *generations*.
4. ***Subtype polymorphism***, another fundamental principle of OOD, the Greek translation of polymorphism is ‘*having many forms*’. In practise it is this mechanism that enables base class inheritance; as all sub-classes derived from the same parent class(es) will be related by their shared parentage.

- 
5. ***Method overloading and overriding***, this allows any derived sub-class to hide any method inherited from the parent class(es) by implementing methods using the same method name as in the parent class. The difference between them is overloading specifies methods with different parameter lists to the parent, the parent method will still directly visible, or by overriding where the parent method is specified with the same parameter list, thereby completely hiding the parent method, although it can still be indirectly accessed.
  6. ***Virtual interface***, the mechanism by which a set of virtual methods are defined by a base parent class, this creates an interface that enables any sub-class derived from a virtual base class to be directly called in the program using a parent class pointer or reference; the difference being that the method interface implements code that is specific to the sub-class definition.

### 3.6.1 Problem space representation in a computational environment

This aspect is almost entirely problem domain dependent, representing almost any real-world problem in a computer inevitably means that some form of compromise will be required. The end results can only ever be considered as a model of the most salient aspects represented in such a way as to allow the evolutionary mechanism to perform the crossover, mutation and evaluation for fitness functions to be performed effectively. *The concept of the latter can be defined in the following requirements.*

1. ***A chromosome data structure*** to hold sufficient information in an appropriate format to allow the problem space to be directly represented in the simplest way possible, whilst still allowing the logic of the problem space to be super-imposed upon it. The chromosome is generally an array, or list, of discretely indexed elements (or alleles in the terminology of genetics), each being analogous to a single letter representing one the four nucleic acid bases (ie. ACGT) that make up the genome of living organisms. Chromosomes can also contain gene sequences that exist across a number of the discretely indexed alleles, ultimately this is a problem space dependency. The very simplest chromosomes consist of an array of binary digits, the interpretation being as on/off switches applied to some aspect of the logic encoded in the problem space algorithm. It is also quite common for chromosomes to have a complex structure that contains more information but also some level of implicit problem space logic itself.
2. ***A set of appropriate crossover and mutation operators*** to perform the generation and modification functions required to produce the new offspring for the next evolutionary generation. These not only have to be closely related to the chromosome representation in order to access the information encoded within, they are also quite likely be dependent on the inter-dependencies between alleles or genes as defined by the problem space. The operator interface can be extended to apply heuristic rule based pressure to influence, or limit, the effects of the stochastic mechanism implicit in all evolutionary algorithms.

### 3. BACKGROUND

---

3. *Encapsulating, (modelling), the problem space logic* within the software that implements the optimisation algorithm. In the simplest systems this is likely to be entirely contained within the objective fitness function used to determine the suitability of the candidate solutions. For more complex algorithms this may include a set of heuristic rules to apply the problem specific objectives.
4. *An objective fitness function* configured to encapsulate the salient aspects of the environment that is modelled in the problem space to enable a quantifiable ‘result’ to be derived, either as a discrete and definable target value or a best possible value, (or range of values), where a specific target value cannot be reliably defined due to complexities within the problem space itself.

This section is the very simplest of introductions to what is a complex and wide ranging subject, in particular the intention has been to present the information in as context-free way as possible. The software in chapter 4 describes the EA problem space design and implementation used for this work in more detail.

#### 3.6.2 The object-oriented design process

The process of modelling a real-world problem in a computational environment using the object-oriented paradigm is usually one of structural and functional analysis, and the subsequent simplification; the process continues until a coherent and functional structure has been determined that defines the salient logic of the problem to be represented with adequate detail, but without being over-simplified. If successful the end result of this process should be a set of functional definitions that define the significant process stages and a structured set of inter-process connections; when considered together these define the overall logic of the problem being represented.

This process of modelling the problem is directly supported by the basic premise defined in object oriented design; the salient logic of a generic problem can be represented in a computational environment by a structured hierarchy of class definitions. The hierarchical structure and the class definitions are related as the connections between the classes define the overall functional logic of the design entity representing the problem.

#### 3.7 Chapter summary

In summary the main objective of this chapter in introducing the background themes as largely independent entities was to provide the basic information to establish a suitable context for the work of the thesis. It is when the discrete subjects are considered in combination that a more complete idea of the real-world practicality of the proposed system can be gained. This was considered to be particularly useful as this assists greatly in establishing a link between a purely abstract software model and the practical use that the results of the research may indicate and possibly even fulfil.

## Part II

# Problem Description



## Chapter 4

# Software model design

The objective of this chapter is to describe the research subject addressed in this thesis by building on the background information introduced in the previous chapters. This information is combined into a working abstraction of a real-world system which is subsequently translated into a software based modelling system.

The first section of the chapter takes the concept of the existing commercially available single-element upper-airspace monitoring system introduced in sections 3.4.4 & 3.4.5, (the Radio Sonde and Drop Sonde respectively). The monitoring and wireless communication sub-systems that characterise these devices form the inspiration for a novel *3-dimensional multi-element clustered duplex wireless linked monitoring system*. The proposed monitoring system has the design potential to dynamically monitor an upper-airspace in 3-dimensions by generating a number of spatio-temporal data streams comprising monitored data from the 3-dimensional zone covered by the dispersal of the discrete elements in the cluster; the common time and space aspect, in effect, will define an implicitly linked data-set for all elements in the cluster.

The mode of operation for this novel system demands an equally novel method to track the dispersed elements in order to determine their physical positions in 3-dimensions in the upper-airspace deployment zone. This aspect has been investigated through the development of a computer model designed to encapsulate the essential aspects of a real-world upper-airspace in which the multi-element monitoring system devices can be represented and physically manipulated in 3-dimensions (four dimensions as time must be included).

It is proposed that the complexity inherent in tracking and positioning these elements in the ‘*correct*’ relative distance magnitude 3-dimensional geometry that the search space will contain so many potential candidate solutions that it will be a problem defined as Non-deterministic Polynomial Hard (NP-Hard) as defined by (Garey & Johnson, 1980) [44], and (Kleinberg and Tardos, 2008) [73], amongst others. It is therefore likely that the use of optimisation algorithms is the only realistic method for generating suitable

## 4. SOFTWARE MODEL DESIGN

---

3-dimensional solutions on a realistically short time scale.

In the latter section of the chapter the general principles of the object oriented design method are introduced as the development basis for the concept of the Virtual Base-class Framework (VBF); this framework comprises a set of software base-classes that, in combination, form a software run-time entity that encapsulates the fundamental requirements of a generalised optimisation algorithm system. These requirements include the underlying iterative processing structure, overall ‘*modelled*’ environment (the 3-dimensional upper-airspace), allele (individual ‘*Sonde*’ element), chromosome (the element cluster), objective fitness function, mutation/crossover operators, and data collation/results sub-systems.

The concept of using the ‘*virtual interface*’ structure such as those supported by C++<sup>1</sup> have been used to implement a wide range of software frameworks designed to target specific problem spaces. A good example of a library specifically developed for Genetic Algorithms is *GA-Lib* by (Wall, 1996) [132], other usage examples of this is the C++ environment developed to model embedded signal processing systems by (Winograd & Hamid Nawab, 1995) [140], and a set of marine geospatial ecology tools using this techniques has been developed by (Roberts et al, 2010) [102]. A generalisation system implemented using the C++ framework of ever increasing importance is that for petabyte data storage, statistical analysis and visualization by (Antcheva et al, 2009) [4]. Two other interesting research studies reflect the evolutionary process from the natural world through to the purely technical, the first is the C++ class library for evolutionary genetic analysis by (Thornton, 2003) [114] and the second is that implemented for evolutionary computation through the use of ‘*design patterns*’ in the Open BEAGLE system by (Gagne & Parizeau, 2002) [42].

Finally there is an overview of the Software Derived Classes (SDC) that have been implemented using the VBF derived software model. The entire software system is fairly complex, as a result this is examined at a relatively high-level to demonstrate the essential derivation hierarchy and fundamental functional paths required to build a generalised iterative optimisation algorithm system. The SDC, when taken as a software entity, essentially maps the concept of a multi-element Drop Sonde cluster to a suitable computer representation, the result is a set of derived classes that form the basis of all simulations used in this thesis.

### 4.1 The real-world upper-airspace abstraction

The objective here is to describe the practical aspects behind the concept of the *multi-element clustered wireless linked monitoring system* and the 3-dimensional upper-airspace environment in which such a system would operate. This is followed by an abstraction of the 3-dimensional concept by simplifying the essential aspects in order

---

<sup>1</sup>This concept is common to most programming languages based on object oriented methodology.

---

to assess the overall problem search space complexity. At this stage this is as a simple description, it is covered in more detail in the software implementation section 4.2.

The ‘*standard*’ Radio Sonde system consists of a single Sonde device (for general weather forecasting), or a Drop Sonde (for weather/climate research), a receiver and data collation station, and a means to track the Sonde whilst deployed. Whichever type of Sonde is used the tracking device (ie. Radar) logs the data defining the track taken in ‘*flight*’ in real-time; this is effectively a 3-dimensions entity as the tracking data-set comprises of triples of bearing & elevation (in degrees), and distance magnitude.

The environmental parameters are monitored, locally stored, formatted into a packet based data-stream, and then transmitted to the receiving station every few seconds. This is subsequently post-processed to produce a combined spatio-temporal data-set by the addition of the location data generated by the tracking system; wind speed and direction data can now be inferred by geometric calculations based on the ‘*delta*’ between consecutive tracked bearing, elevation, and distance magnitude data samples.

If the Sonde system were to be augmented by the addition of a number of discrete, independent, Sonde type elements then the resultant data-set from all these elements could be combined into something that resembled a 3-dimensional data-set; this would have the potential to contain a detailed spatio-temporal ‘*description*’ of current events in the monitored upper-airspace. This type of data-set has the potential to ‘*map*’ the internal meteorological parameters inside a storm cloud (for example) that, after post-processing, is effectively a near real-time modelled representation that may have potential as a climate research tool; the data-set could be used as part of the larger climate modelling process or simply replayed in a graphical display.

The practical and software modelled realisation of a ‘*multi-element Clustered Drop Sonde*’ is a real possibility by the application of the ‘*Smart Dust*’ system described in appendix A, and through the development of a software system to model the element cluster employing a form of optimisation algorithm. This forms the basic premise of the research question behind the work of the thesis.

#### 4.1.1 The Clustered Drop Sonde hardware abstraction

The hardware and software system has been designed to model the real-world system using Smart Dust (SD) motes and associated communication and networking hardware as the source of the abstraction.

The standard Radio and Drop Sonde system, introduced in section 3.4, is used for upper-airspace monitoring by meteorological organisations worldwide, as an everyday forecasting tool, and for many different aspects of meteorological and climatological

## 4. SOFTWARE MODEL DESIGN

---

research.

In summary, these devices are self-contained miniature weather stations designed to monitor upper-airspaces that are ‘*flown*’ aloft by helium filled weather balloon or by being ‘*dropped*’ from an aircraft in the case of the more esoteric Drop Sonde.

It is the Drop Sonde introduced in section 3.4.5 that is the specific model used as the motivation of the work of this thesis, although either would suffice as they have many technical similarities.

The data types generated by what might be described as the *standard* meteorological Sonde, are shown in two lists in section 3.4.2. The first lists the data types directly monitored through on-board sensors, the second lists the data types that can be deduced from post-processing the tracking data that describes the Sonde *flight path*. The tracking data is often generated by Radar tracking, but can be generated by Global Positioning System (GPS) modules fitted on certain types of Radio Sonde designed for specific monitoring purposes, those used for general meteorological monitoring are not usually fitted with GPS. The single synoptic data stream produced by any single element Sonde device may be considered as a limitation, but one that can be improved upon by using a number of the standard Sonde devices simultaneously in a cluster spread over a defined area of interest. The simultaneous deployment method can, in theory, be used with any existing commercial Sonde device; there are technical challenges to implementing this as a practical system, specifically, in providing sufficient independent radio channels for the data communication and in tracking and uniquely identifying each Sonde device simultaneously and consistently. These are practical challenges, and although they are not insurmountable using existing tracking technology, it is implicit in the research subject to investigate the development of alternative and novel methods of achieving this.

There are several aspects of the SD system that can be used to great advantage in the design of a Multi-element clustered Drop Sonde (MCDS), specifically those of, low cost, light weight, and compact dimensions. It can be expected that the cluster elements will be sympathetically displaced by air currents, thereby revealing information about the momentary wind speed and direction and the rotational velocity of these air currents, something which the singly deployed standard Drop-Sonde is not equipped to do.

The desired mass and size of an SD based Drop Sonde element is similar to the average hail stone of around one and three quarter inches (45mm), and around an ounce (25 grammes) in weight. Hailstones are formed by being trapped for a period of time within the circulatory air currents frequently found in storm clouds as researched by (Wang and Chang, 1993) [134], (Jouzel et al, 1975) [69], and in particular the work on supercooled water by (Brownscombe & Hallett, 1967) [12]. It may be expected therefore that if an SD element were to be configured as a simple Radio Sonde and of similar size and mass to a medium size hailstone (anything over one inch (25mm) is considered severe) [88], then it can also be expected to behave in a similar manner when subject

---

to the same internal forces as a hailstone; in this way the SD element would directly ‘*experience*’ the same forces and therefore be able to effectively monitor them.

The SD elements are configured to operate as a cluster to produce a time dependent dataset *picture* of synoptic parameters and data directly relating to their position at any given instant. The resolution of the data ultimately depends upon the number of SD elements in the cluster and the effectiveness of the reconstruction software model; the result will be a near real-time 3-dimensional representation of the dynamics inside the upper-air zone.

This encoded geometric data is post-processed by a dedicated localisation reconstruction computer model to give a 3-dimensional representation of the SD element dispersal geometric configuration of the upper-air zone.

#### 4.1.2 An overview of a clustered Drop Sonde

The objective of this section is to describe how a Drop Sonde cluster may operate in a real-world deployment, with particular design emphasis on mitigating certain practical constraints pertinent to the wireless transmission environment.

There are a number of practical requirements, perhaps, the most significant is the development of a design strategy to avoid synoptic data transmission corruption during the inevitable transmission clash between two or more Radio Sonde cluster elements and all other elements, including the dedicated ground station.

The notional 3-dimensional Drop Sonde Cluster has two distinct, but closely related sections, the dynamic group of SD elements that constitute the airborne Drop Sonde cluster and the cluster interface element (the dedicated ground station, or static gateway) that receives the data streams from the SD cluster; in the Crossbow™ system the static gateway device could be any of the interface devices described in appendix A.2, the ideal device is the Stargate™ computer based interface described in appendix A.3.

The static gateway provides the network connection to the *outside world* to any external processing systems through which the data streams produced by the cluster elements can be collated, stored and post-processed. As the name suggests the static gateway is at a fixed position that remains relative to the coordinate frame as dictated by this physical location. If this coordinate reference frame location is also fixed, as is the case for a terrestrial ground station, then the airborne cluster and the static gateway can be said to have a common coordinate reference frame; all positional data generated by the duplex communications will be relative to this and no further coordinate frame correction will be required.

If, however, the location of the static gateway is aboard an aeroplane in flight, as is usually the case with a Drop Sonde receiving station then the coordinate reference frames cannot be considered as being coincident. In this situation additional positional

## 4. SOFTWARE MODEL DESIGN

---

information from a GPS onboard the aeroplane will be required to define a shared common reference coordinate system to provide the positional validation. The concept of the multiple element Drop Sonde based on a cluster of Smart Dust devices can be summarised thus.

- It extends the capabilities of the *Standard* (single element) Drop Sonde
- Comprising a number of cluster of wireless linked, mesh networked elements
- The Cluster disperses into a 3-dimensional formation on release
- Each element monitors its own local airspace
- Results are sent to gateway using a synchronised communication algorithm
- The combined dataset is post-processed to produce time dependent data
- High resolution directly measured synoptic parameters
- 3-dimensional relative positional data encoded as time dependent RSSI data
- Vertical wind speed/direction, rotational wind fields encoded in the RSSI data

### 4.2 The software modelling system

To implement the geometric reconstruction requires a method by which freely distributed SD cluster elements can be represented in a computer environment such that they are located in 3-dimensions, with acceptable positional accuracy, and with a defined time-stamp. The proposed solution is to dynamically model a 3-dimensional air-space in which free-flying cluster elements are *released* and manipulated in all dimensions to generate geometric configurations that may be tested for ‘*correctness*’ against an input data-set.

#### 4.2.1 Clustered Drop Sonde run-time overview

The only real-time data available for the positional reconstruction is the Radio Signal Strength Indication (RSSI) data automatically generated by the SD elements as they communicate via the duplex wireless system. This data is modelled by having each element transmit data in strict element-by-element rotation, the data is received by all other cluster elements as they will be in *listening-mode* when not transmitting.

The RSSI data is stored as discrete groups, which when combined as a time-stamped dataset it contains an implicit encoding of the 3-dimensional SD cluster geometry. To summarise, the RSSI data is known to be of inherently poor quality, foremost concerns include the non-linear representation of distance, prone to interference/reflections, antenna orientation effects, transmission collision corruption and data packet loss. Whilst

---

some of these limitations can be mitigated to an extent by design it is still unlikely to be entirely suitable for the accurate reconstruction of the 3-dimensional geometry of a clustered Drop Sonde.

The following illustrate some of the primary run-time stages of a Smart Dust based Drop Sonde cluster.

1. **Cluster initialisation.** Prior to deployment all SD elements in the cluster are powered up, those defined as cluster elements await the cluster registration phase that is initiated by the SD element designated as the static gateway. When all elements have been successfully detected, connected to the mesh network, and have received their unique registered identification code, the networking sub-system distributed across all SD elements automatically constructs the network routing tables. At this point the dynamic cluster has been successfully initialised and is in standby mode with only the static gateway element *awake*.
2. **Cluster monitor mode.** Just prior to deployment the individual SD elements in the cluster are still paused and *listening* for the static gateway to issue the instruction to come out of standby mode and enter into the meteorological synoptic monitoring phase.
3. **Cluster deployment.** The usual mode of deployment for a *standard* Drop Sonde is via an ejector chute from a suitably equipped research aeroplane. This also houses the receiving ground station which, in this example, also contains the static gateway, data collation, and post-processing system. On deployment the cluster of Drop Sondes emerge directly above the upper-air-space that is to be monitored.
4. **Cluster element synchronisation.** Transmission collisions will result in random alteration of the signal strength as received by all non-transmitting elements. As the set of RSSI values that form the data-set from which the software model is to reconstruct the 3-dimensional geometry this is clearly an unacceptable situation. Therefore, in order to produce a valid cluster data-set, the SD elements must operate a synchronised transmission interval pattern that is controlled by the static gateway.
5. **Transmission collision avoidance - round-robin scheduling.** This is one of the very simplest scheduling algorithms, but it does ensure that each SD element is given equal priority as it can only transmit its data packet when its unique identification code indicates that its turn has arrived and it can become active. The SD identification code is a number in the range  $0..(n-1)$ , the elements monitor the transmissions as they proceed, when the identification number that precedes its own is detected then it is required to transmit immediately. In practise a time-out mechanism will also be necessary to allow for any element that fails to transmit.

#### 4. SOFTWARE MODEL DESIGN

---

6. ***The data-set collation process*** The data monitored by each SD element is ultimately transferred to the static gateway, directly as a received transmission, or via the mesh network as described by the routing tables. For the latter the mesh network system automatically forwards the data packets to the static gateway, which is designated as the default remote SD element. The gateway is attached to an external processing and storage device, the Crossbow Stargate™ interface device for example, this has the task of collating the packets into a coherent spatio-temporal data-set, removing the inevitable packet duplication and requesting missing or corrupt packets are resent.
7. ***The instantaneous Data-set structure.*** After an SD element has transmitted all other elements, including the static gateway, will have an RSSI value pertaining to that element; this set of values represents a snapshot that loosely describes the distance magnitude (but not including any directional information) between the transmitting SD element and the receivers.  
When all elements, again including the static gateway, have completed the *round-robin* transmit/receive cycle the spatio-temporal dataset that results consists of a set of magnitude values that represent, in an encoded form, the 3-dimensional geometry of the cluster formation for the interval that covers the length of time that it took for all elements to complete the cycle. The dataset of RSSI values are uniquely identified by the time and identification code fields of the sender and receiver SD elements. It is this data-set that forms the input data-set for the reconstruction software model to use as the basis for the fitness function comparison stage when assessing candidate solutions for *correctness*.
8. ***Transmission calibration.*** The static gateway is also included in the *round-robin* scheduling algorithm as it is also part of the active cluster network as it is the ultimate destination for all data packets generated by the cluster. The role of the gateway element is particularly important as it is likely to be the most distant element from the airborne elements in the cluster and therefore this distance is likely to be the *most similar* for all SD elements.  
This aspect makes the set of RSSI values between the gateway and the cluster elements useful in *calibrating* the signal strength values as each SD element should all produce relatively close values for what is likely to be a similar actual distance. The differences between these values are more likely to represent receiver sensitivity and antenna orientation variations than effects due to distance.
9. ***Cluster demise.*** The length of time a standard Drop Sonde survives is largely dependent on the height of its deployment, given that the rate of descent of a standard Drop Sonde is 20 knots and the deployment height being many thousands of feet this can be easily calculated. For the multiple element Sonde envisaged here a stated descent rate may not be applicable, the dynamics of the upper-air-space being monitored can be expected to have a profound effect on the descent rate of the elements, especially if the air currents within the airspace are strongly circulating, as would be expected within a storm cloud.

---

## 4.3 Virtual base-class frameworks

The concept of the Virtual Base-class Framework (VBF) is of a composite software specification and design entity that utilises the fundamental object oriented design (OOD) principles of inheritance, polymorphism, and method overriding; the resultant interface design ensures that any software derived from a VBF must therefore conform to the original logical structure. The intended use for the VBF is to define the essential logical structure that characterises a class of problems that are functionally or structurally related.

The result is a design for a set of classes which, due to the inherent pure virtual function and class structure, cannot be directly implemented to create a run-time software entity. The VBF provides the implementation basis of run-time software to derive the set of classes to implement the required, functionally related, optimisation algorithm run-time environment.

### 4.3.1 System processes in a virtual base-class framework

This section describes the mechanisms supporting the concept of a virtual base-class framework from the more practical context as represented by the C++ programming language as defined by the ANSI<sup>1</sup> and ISO<sup>2</sup> standards organisations. This language is compliant with the object oriented principles listed above, but it also defines several run-time features that can be used to provide additional functionality to directly support the VBF concept.

1. ***Pure virtual functions*** . With C++ it is possible to declare a class method as being *virtual*, essentially this is just a run-time pointer ***place-holder***, the method itself does not properly exist in the defining class. The virtual declaration defines a function name and optional parameter list that must also be fully declared by any derived class, even though an empty code method implementation will suffice. This mechanism provides the basis to define a *virtual interface* to which all derived classes must adhere.
2. ***Abstract classes (using pure virtual functions)***. Any class with at least one virtual function is defined as abstract, attempting to directly instantiate an abstract object will cause a compilation error. As for the virtual function the entire abstract class declaration is also effectively a run-time pointer *place-holder*. It is quite possible to declare a pointer using the abstract class as its base type, this will not cause a compiler error, but it does allow for that pointer to point, or address, any object derived from that abstract class; this is a form of automatic type casting.

---

<sup>1</sup>American National Standards Institute

<sup>2</sup>International Organization for Standardisation

## 4. SOFTWARE MODEL DESIGN

---

3. *Parent (base) class pointers and references.* In practical terms these are the means by which discrete object instantiations within a software entity can be directly and uniquely addressed, in turn this provides access to the particular method associated with specific object instances. How these are implemented is not defined by the OOD paradigm as this will be programming language specific; for example, C++ defines (address) pointers and references, whilst the Java language specifies only references (for security reasons). Non void C++ pointers have a fundamental type defined by the pointer variable declaration. This type can be derived from any point in an inheritance hierarchy, including the virtual base class definition; this enables any pointer to be used to access any other object pointer derived from a shared base class in a common inheritance hierarchy. The functional aspect to this access mechanism enables any shared methods defined by the shared parent class(es) in the inheritance hierarchy to be accessed by the derived sub-classes.
  
4. *Run Time Type Identification (RTTI).* This is an alternative method of casting an object to a compatible pointer type as described above with a virtual function/abstract class method. This method is commonly referred to as *late binding* as it occurs during the run-time process, it is also quite expensive in terms of CPU clock cycles as address/pointer redirection is normally necessary in order for the run-time system to determine exactly which object type in the inheritance hierarchy is active, the level of method overriding or overloading, and then to determine the exact version of the method required. It has not been used in the mechanism underpinning the VBF, but has found other uses in the software system. Although sub-type polymorphism (introduced in section 3.6) is the fundamental operational mechanism behind the VBF, the techniques described above are all required to make the VBF function as intended.

### 4.3.2 A practical virtual base-class framework

The object oriented design paradigm directly supports the concept of an abstract or virtual base-class framework. This section builds on the concepts introduced in the previous section to define a virtual class hierarchy that encapsulates the common run-time functionality commonly required in a generalised, single, multi or many-objective iterative optimisation software system.

The section depicted as green in figure. 4.1 indicates class definitions that are still part of the VBF section that have been declared independently to enable the necessary facility of discrete derivation according to the requirements of a particular algorithm. The yellow sections indicate a set of derived classes that fulfil the requirements of the virtual function interface, in so doing they define a complete specific optimisation algorithm that can be instantiated as a run-time system.

---

The overall intention of the diagram is to illustrate the system in high-level terms as to the function call structures and variables types, except where they are required. The actual VBF interface is rather more complex than this diagram shows, but it is sufficiently detailed to demonstrate the main VBF interfaces and how these are translated into externally defined run-time classes that must be directly derived from the set of abstract classes.

The virtual interfaces for the fitness, operator and chromosome class definitions are described in this chapter, the process of using these abstract classes to produce the first set of derived classes that are used in the simulations is detailed in a subsequent section. A representative list is shown below.

- Initialisation
- The iterative run-time structure
- Algorithm interface
- The allele data structure
- The chromosome data structure
- The chromosome population data structure
- Mutation & crossover operators
- The default objective fitness function
- Run-time compound data collation sub-system
- Results generation and output

Diagram 4.1, shows an overview of the main classes, interfaces, and the structure of the overall of the VBF depicted as the clear sections. It is intended to combine class hierarchies, fundamental data connections, and functional interfaces, this differs from ‘*standard*’ class diagrams as the principle purpose is to depict a complex level of information in a single, relatively simple diagram.

#### 4. SOFTWARE MODEL DESIGN

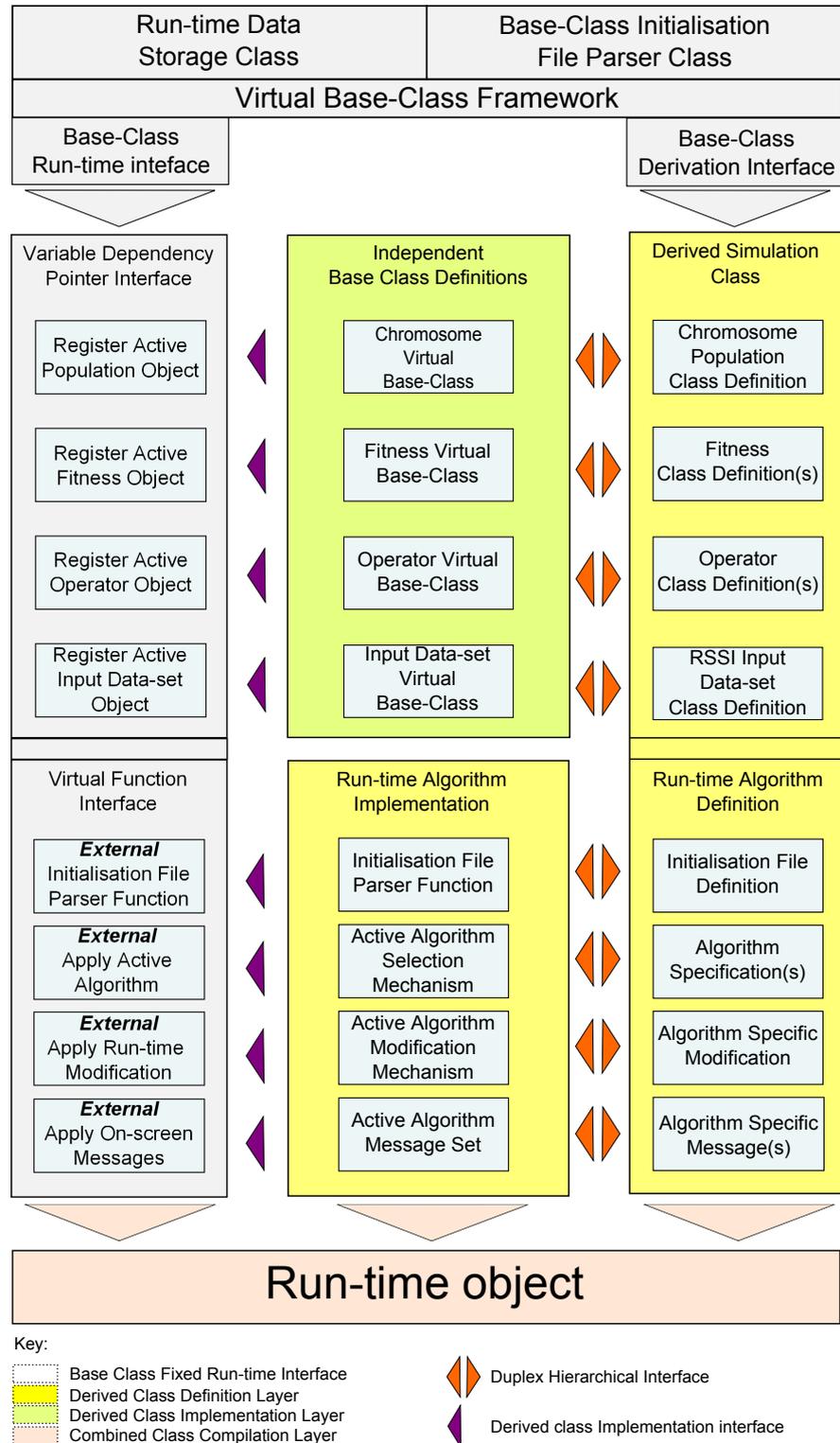


Figure 4.1: The Virtual Base-class framework block diagram

---

## 4.4 Implementing a geometric reconstruction model

The objective for this section is the high-level introduction and description of the 3-dimensional geometric reconstruction modelling system used for all simulations in the thesis. A software model has been developed based around the concept of a generalised functional framework principally influenced by the techniques defined in the object-oriented design paradigm that was introduced in section 3.6.

One frequently occurring question in computer science is how to represent a particular problem in a computer system in such a way that the essential characteristics are encapsulated with sufficient detail beneath which a model would cease to be an effective representation. A logical first step is to examine the problem space to determine which features represent these essential characteristics, and, from this state to establish an appropriate method of representation in a computer. For the purposes of this work the problem space specification can be broken down into a number of contextual areas for consideration. They can then be further defined in terms of boundaries, definition or required functionality.

Each of these short sections examines the problem from an abstract viewpoint in order to define the minimum requirement to fulfil the purpose of that context. The software implementation of each abstraction is likely to be considerably more complex however.

- The complexity of the search space
- Representing the problem space in a computer
- The algorithm specification and control interface class
- The allele: the fundamental dimension definition
- The allele: **Drop-Sonde** element class
- The n-dimensional upper-air zone
- The chromosome: **Drop-Sonde** element cluster class
- The chromosome: population class structure
- The operators: the virtual class structure
- The objective fitness: 3-dimensional element calculation class
- The Data storage, collation and output sub-system

## 4. SOFTWARE MODEL DESIGN

---

### 4.4.1 The complexity of the search space

There are many real-world combinatorial problems that can be considered as appropriate for the application of optimisation algorithms as a suitable means of searching for, or creating, suitable candidate solutions; in many situations the application of an optimisation algorithm may be the only realistic means of generating any sort of solution. Frequently problems of this type are designated as being Non-deterministic Polynomial Hard (NP-Hard), as it cannot be guaranteed that a plausible or general solution exists in some form, nor that it can be calculated within a polynomial or otherwise usefully brief time-frame. The general form of the type of iterative NP-Hard problem considered here are those where the number of possible candidate solution permutations that *may* exist for any given problem is so large that it is not possible, or even realistically feasible, to iterate through all of them to determine the best possible solution, let alone find an optimal solution (this assumes a solution exists of course).

One such problem that may be considered as a candidate for inclusion in the NP-Hard problem class is the process of resolving relative 3-dimensional positional geometries that describe the a cluster of discrete and unrestrained elements using a data-set comprising a series of indeterminate inter-element distance (magnitude) values; by definition, such a data-set must be devoid of any Cartesian or polar (vector) referencing information by which this data-set can be considered as geometrically determinate. This implies the need for a systematic approach to establish a means by which each discrete element may be located, with adequate precision, in a suitably referenced 3-dimensional space, and within a defined time frame. The requirement to know the location of the discrete element is an independently valid and complex objective, in the environmental monitoring system context this also provides the necessary localisation information to validate any monitored data in time and space; without this information such data would be of little practical use.

This problem is essentially one where the problem space is searched for suitable candidate 3-dimensional geometric solutions that can be considered as being somewhat analogous to a *fully connected graph*. This sort of connectivity is similar in a number of ways to the process of *Very-large-scale-Integration* (VLSI). This process of evolving integrated circuits by combining thousands of electronic components onto a single chip is also one where many interconnections (edges) must be made between nodes, it is an area to which the optimisation type is ideally suited; this is the subject of many research papers that cover many different applications covering a wide range of fields as described in section 2.4.1. The complexity of the search space being modelled is one where a large number of geometric permutations are contained, although not infinite due to the resolution of a 32 or 64 bit computer. This is defined by the IEEE Standard for Floating-Point Arithmetic (IEEE 754) as shown in figure 4.2.

---

Level	Width	Range	Precision
Single precision	32 bits	$1.18 * 10^{-38}$ to $3.40 * 10^{38}$	(Approx. 7 decimal digits)
Double precision	64 bits	$2.23 * 10^{-308}$ to $1.80 * 10^{308}$	(Approx. 15 decimal digits)

Figure 4.2: Simple search space complexity calculation

The **Precision** data shown in figure 4.2 implies a minimum numeric resolution that can be reliably reproduced by a generic computer system. In terms of the number of incremental steps available this will ultimately depend on the volume or area of the modelled space and the number of bits used by the processor. in the simplest case above, of 32 bits, this equates to a smallest step size of 0.0000001 that can be accurately resolved by a processor with minimal rounding error.

Although the model uses double precision numerical representation, the usual rounding error under the C++ programming language is compiler dependent, but is commonly six digits of precision; however this is still more than adequate given the indeterminate nature implicit in the RSSI based input data-set will probably swamp any errors introduced in this way.

Example 1: A simple problem space (time is not considered):

Where:

X, Y & Z Bounds	10.0
Minimum step size	1.0
Elements	10

Possible solutions  $(X * Y * Z)^{elements} * 1.0/Stepsize$

Permutations  $1000.0^{10} * 1.0 \approx 10^{30}$

Figure 4.3: A more complex and representative search space complexity calculation

As can be seen in the trivial example of figure 4.3 the number of permutations that even this calculation gives is very large, ( $10^{30}$ ), this particular number has its own order of magnitude name of ‘a quintillion’ (or ‘nonillion’ on the short scale). To put this number into a real-world context, the number of bacterial cells on the Earth is thought to be approximately  $5 * 10^{30}$  (Whitman et al, 1998) [138], this is a very large number that encompasses terrestrial flora & fauna in its entirety. The result from a simple

#### 4. SOFTWARE MODEL DESIGN

---

example equates to a fifth of that value may be considered as a clear indication that the number of potential solutions in the problem space is potentially very large.

Example 2: A more representative problem space (time is not considered):

Where:

X, Y & Z Bounds            1000.0  
 Minimum step size        0.0000001  
 Elements                    10

Possible solutions  $(X * Y * Z)^{elements} * 1.0/Stepsize$

Permutations     $(1000.0 * 1000.0 * 1000.0)^{10} * \frac{1.0}{0.0000001} \approx 10^{90}$

Figure 4.4: Simple search space complexity calculation

The example in figure 4.4 is closer to the search space dimensions that has been used in all the simulations used throughout the thesis. The only significant differences are the dimensions of the bounded zone and a more representative step size. These simple changes have resulted in significant change to the number of possible 3-dimensional permutations geometries, a difference of 20 orders of magnitude. There is one other notable difference however in that the above examples are limited to 10 elements, whereas the majority of the simulations used a range of 10 to 200 elements.

Problem search space size (cube 1000.0, mutation step of 0.0000001)										
Cluster size	5	6	7	8	9	10	12	15	20	25
Combinations	$10^{45}$	$10^{54}$	$10^{63}$	$10^{72}$	$10^{81}$	$10^{90}$	$10^{108}$	$10^{135}$	$10^{180}$	$10^{225}$
Factorial	120	720	5040	$4.1^4$	$3.6^5$	$3.6^6$	$4.7^8$	$1.3^{12}$	$2.4^{18}$	$1.5^{25}$

Table 4.1: Approximate geometric permutations against cluster size

Table 4.1 further illustrates the search space complexity for a number of element sizes up to 25<sup>1</sup>, these numbers are truly astronomical in size; those numbers highlighted in red exceed a Googol ( $10^{100}$ ) (Origlio & Weisstein, 2012) [93]. The table has a second row of numerical data representing the **Factorial** combinations for the cluster sizes, these do not consider the mutation step size as they are just the combinatorial permutations possible for the cluster size. As can be seen the number of permutations

<sup>1</sup>It was not possible to complete the set of element sizes up to 200 elements due to the limitations inherent in the computer used when handling such large numbers.

---

clearly becomes considerable in itself, the numerical data in the **Combinations** row being clearly greater than  $n!/2$  for  $n$  loci; hence this can be considered an NP-hard problem (Mester et al, 2003) [84], an assertion also supported by the general class of multidimensional knapsack combinatorial optimisation problems (MKP) as also being an NP-hard problem (Li et al, 2006) [79].

To further (empirically) illustrate the time required for an exhaustive search of every possible permutation, the assumption is that one complete iteration of the model software running an Evolutionary Algorithm simulation could produce a single geometric candidate solution, in one micro-second, (this is not currently possible). Such a computer system would take  $10^9$  seconds to complete the example in figure 4.3, for the more complex example in figure 4.4 the time required would be  $10^{84}$  seconds to complete the exhaustive search. These numbers equate to 31.7 and  $3.2^{76}$  years respectively, the current acknowledged age of the universe is 13 billion years, or a mere  $13 \times 10^9$  years.

These numbers suggest that problem such as the one under discussion can only, realistically, be tackled using an optimisation algorithm type model, such as an Evolutionary Algorithm.

## 4.5 Representing the problem space in a computer

This section describes the simplified representation of a 3-dimensional upper-airspace environment, the Multi-element Clustered Drop Sonde (MCDS), and the wireless enabled receiving station; the latter may be ground based or part of the airborne system. The common factor with all environments modelled in a computer system is the need to reduce the target environment to a set of the most relevant characteristics, simultaneously retaining sufficient detail to ensure the model remains a reasonable and practical facsimile.

Whilst this process must include the physical environment, defined here as the underlying 3-dimensional structure, but it must also define the required interfaces for the dynamic manipulation of the environment in all relevant aspects; those appropriate to the requirements for this search space are listed in figure 4.5, and shown graphically in figure 4.6.

The stages listed in figure 4.5 all form part of this section and are described in the idiom of their software implementation. In this section the simple objective of this section is to introduce these design stages as part of a high-level illustrative process. The format of the model selected obviously follows the requirements of a 3-dimensional modelled zone in which a set of entities can be presented in free positions within that zone, the structure chosen must also allow for alteration of those positions and for a means by which they may be positionally referenced without ambiguity.

#### 4. SOFTWARE MODEL DESIGN

---

- An overall data structure integration strategy
- An overall system integration strategy
- A 3-dimensional upper-airspace data structure
- A 3-dimensional cluster element data structure
- A defined point of reference for all axes and dimension ranges
- A set of Defined data input and output run-time data structures

Figure 4.5: Simplified overview of the stages in the design process

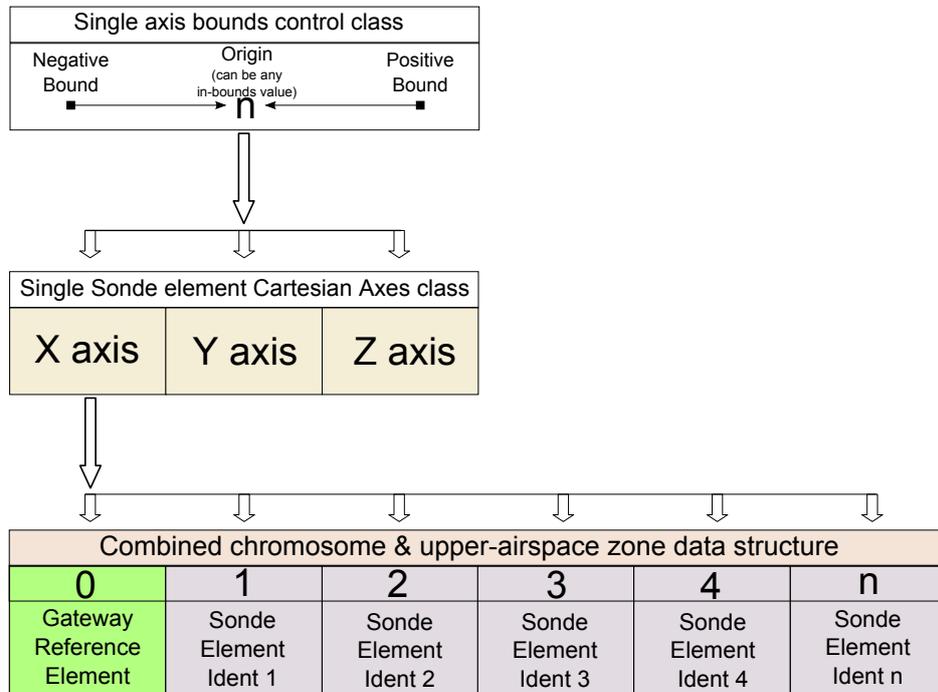


Figure 4.6: Upper airspace and cluster element overview

Put another way, the 3-dimensional positions of the entities can exist as relative data but there must also be a means by which this can be referenced to a common origin within a known, and defined, frame of reference.

The image in figure 4.6, whilst appearing simple actually illustrates virtually the entire modelled environment in a straightforward and hierarchical way, all of these entities are covered in more detail in latter sections. The top part of the figure depicts the upper

and lower bound definition class that defines all axes used in the model,<sup>1</sup>, the middle part of the figure describes the essential aspects of the individual cluster Drop Sonde elements definition class, (including the static reference gateway element), the lower part of the figure illustrates the chromosome array/vector structure; the latter also forms the basis of modelled upper-airspace zone object as it encapsulates the MCDS elements zone and also provides the essential frame of reference as it defines the local origin, (which can also be an absolute origin), and the upper and lower bounds that define the overall 3-dimensional ‘*upper-airspace*’ zone.

In actuality it is also derived from from the same cluster Drop Sonde element class which allows it to be a moveable element within another external frame of reference, (and therefore all encapsulated elements move with it); which is a necessity if the gateway element (always mounted at ‘*an*’ origin) is mounted on an aeroplane in flight.

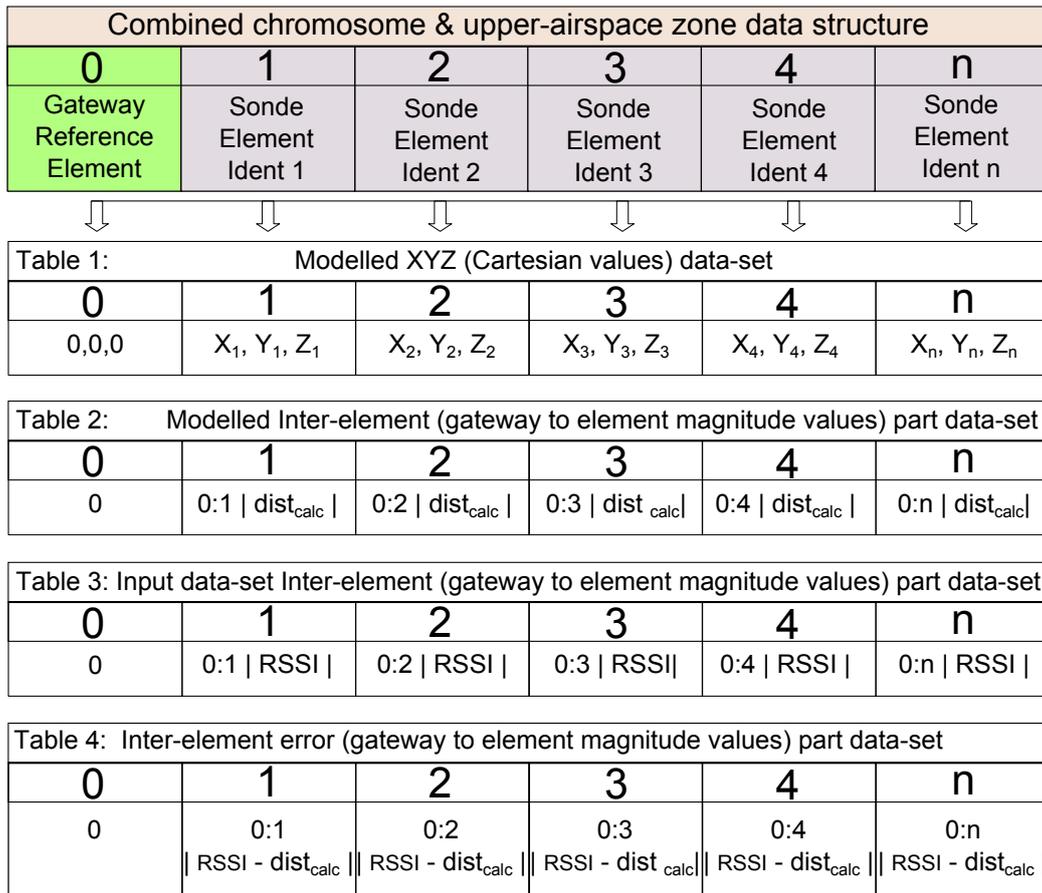


Figure 4.7: Upper airspace and cluster element overview

<sup>1</sup>All 3-dimensional entities must ultimately be derived from this class to ensure dimensional consistency in the frame of reference

## 4. SOFTWARE MODEL DESIGN

---

Figure 4.7 illustrates the principal stages (Tables 1 — 4) in the iterative process leading to the generation of the 3-dimensional data required for the objective fitness function. Table 1 represents the XYZ Cartesian triples (relative to the local gateway element) generated by the modelling software, these are then translated into distance magnitude values between the discrete cluster element and the gateway element (Table 2)<sup>1</sup>. Tables 3 & 4 depict the input data-set RSSI distance magnitude data values, (again between the gateway and discrete cluster elements), and the distance magnitude difference values respectively. The latter data are then summed (in the objective fitness function) as an overall magnitude fitness value. The raw Cartesian magnitude data is generated throughout all runs and locally stored in the cluster element objects and in the overall chromosome data structures, they are not currently output as they are not required at this stage in the development of the system. An example objective fitness and statistics output data-set (as in Table 4) is included in appendix E.

### 4.5.1 The algorithm and run-time control

The operation of the software model is largely controlled by the user settings in the initialisation file as listed in appendix C. In design terms this aspect is fundamental to the operation and internal structure of the VBF and also therefore any classes derived from it, as a direct result the combination of the iterative aspect of the VBF.

They are introduced here as they are part of the general algorithmic interface and, in terms of the model software control aspect, common to all of the run-time models used in the thesis.

<b>Parameter:</b>		<b>Operational Algorithm Type</b>
Steady state	0	Single chromosome replacement EA
Part generational	1	Chromosome(n) replacement EA
Generational	2	Chromosome population replacement EA
Random (Progression)	3	Random algorithm - builds on previous iteration
Random (Reset)	4	Random algorithm — re-initialises every iteration
Hill-Climber	5	Simple stochastic hill-climber

### **Parameter: Simulation run & stop condition**

maxIteration	10000	Maximum iterations performed (if useStopFitVal = false)
minIteration	1000	Minimum iterations performed (regardless of stopFitVal)

---

<sup>1</sup>In this example, for clarity, only the gateway to element data points are shown, the discrete, paired, inter-element data points (ie. 1:0, 1:2, 1:n etc.) are required to complete the data-set

---

useStopFitVal	0	False (0), True (positive/negative value)
stopFitVal	5.0	Stops run if fitness $\leq$ stopFitVal (minimising fitness)

#### 4.5.2 The allele: the fundamental dimension definition

The modelling system requires a base numeric type that defines the fundamental precision, this may be a simple integer or, as in this case a real number as defined in figure 4.2. For the VBF this is defined as being a **template type**, this does not define any specified type as this is deemed to be algorithm or implementation specific.

Therefore the actual numeric base type must be specified by the deriving class as the point of specification to be semantically and syntactically correct, the template is shown in a significantly abridged form in the following list.

```

template <typename datType_B>
class DataPoint_vB // Abridged
{
    datType_B    datInit;    The initial position
    datType_B    datVal;    The value associated with this Line Point
    datType_B    datValMin;  The Notional minimum bound
    datType_B    datValMax;  The Notional maximum bound
    datType_B    datOrigin;  The Notional origin
    datType_B    absRange;   The Absolute value -i, datValMax - datValMin
}

```

To use this template to derive a basic fundamental type is straightforward, this defines a class, **Point\_D**, that defines a class directly from the template of type **double** (precision), and pictorially in figure 4.8.

#### The derived fundamental dimension class

```

class Point_D : public DataPoint_vB<double>

```



Figure 4.8: Euclidean magnitude between discrete element pair

#### 4.5.3 The allele: Drop-Sonde element class structure

At its simplest, the computer representation of an individual Drop Sonde element could be a 3-tuple of real numbers that represents a Cartesian position in the X, Y & Z axes.

## 4. SOFTWARE MODEL DESIGN

---

For the current model specification there is no requirement to provide any means for the Drop Sonde representation to store or process the environmental synoptic data as the model is only concerned with reconstructing the geometric positions of each Sonde element as a 3-dimensional cluster in a suitable time-frame.

The Cartesian X, Y and Z axis tuple data structure is defined as a class that encapsulates a 3-tuple of 4-tuples, the latter are instances of the **Point\_D** class objects described in the previous section, the 3-tuple is a simple C++ vector with additional direct access pointers to each defined axis. The 4-tuple Cartesian data structure therefore defines the axis range as the notional minimum and maximum range values, the origin and the 3-dimensional point positional data within the respective bounds. The angle between these axes is always assumed to be orthogonal to give the standard XYZ Cartesian construct and therefore not stored implicitly. This is shown diagrammatically in figure 4.9

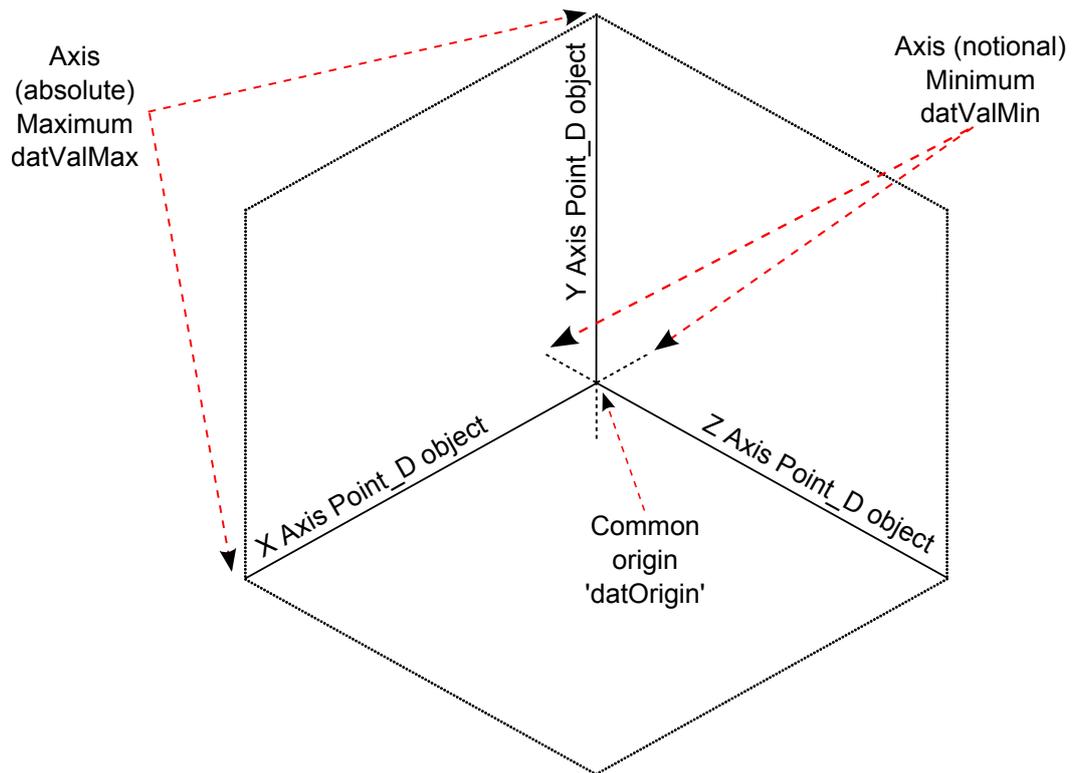


Figure 4.9: VBF n-dimensional zone (Drop Sonde) representation

Essentially this is all that is required, along with the expected set of function methods and control parameters, to represent a simple 3-dimensional entity as a discrete object. However there is also the requirement for a referential context to give the 3-dimensional

---

position contained in the tuples the necessary validation reference frame, without which the element cannot be *placed* or share the space with any other element. This is addressed in the next section.

#### 4.5.4 The n-dimensional bounding zone and sonde element classes

The **upper-air zone** could be represented as a simple n-tuple of discrete numbers that define the upper limit of each bounding axis for a cuboid space, or boundary restraint parameter in the case of a sphere, cylinder or other more complex volume space. The volume space defined for this work is a simple cuboid, this configuration is quite sufficient to represent a modelled 3-dimensional airspace that just has the primary task of representing a bounded zone inside which other elements can be *contained*.

In common with the Drop-Sonde elements this space is also defined as three individual **Point\_D** class objects to define the X, Y & Z axes configured with a common origin, in this instance this is coincident with the minimum bound values, both are set at zero, again as shown in figure 4.9. That the same figure describes the essential features of both Drop Sonde element and 3-dimensional ‘*upper-airspace*’ zone container is quite logical as, in effect, both the Drop Sonde and the n-dimensional space can be viewed as the same 3-dimensional bounded space from different viewpoints. It therefore follows that the n-dimensional space is directly derived from the same base class as that which defines the Drop Sonde; this gives the n-dimensional space object the necessary Cartesian axis definition to provide the reference frame to define the positions of the contained Drop Sonde elements; essentially the n-dimensional upper-air zone class is a Drop Sonde with extra functionality required to perform as a container class. It also allows the entire n-dimensional zone to be separately defined by some external validation reference frame as a discrete entity; as a direct consequence this also applies to all the Drop Sonde elements contained within and therefore the entire zone can be *positionally* manipulated as a whole if required.

#### 4.5.5 The chromosome and associated population classes

The chromosome in the evolutionary algorithm paradigm forms the fundamental processing unit that encapsulates one or more of the most significant parameters that define the relevant features problem space such that it can be manipulated within the optimisation algorithm system. The format of some of the more common evolutionary algorithm chromosome types has been introduced in section 3.6.1, they often consist of a simple bit-string or an arrayed sequence of characters or numbers with each indexed allele or gene representing some aspect of the problem space, the interpretation of which is defined by the rules encapsulated in the active optimisation algorithm.

The more complex chromosomes types can be configured to represent physical shapes or volumes; as a result to adequately define these may require more information per allele, or chromosome index, than can be represented by a single indexed value. This is

## 4. SOFTWARE MODEL DESIGN

---

the chromosome type that is required for this implementation, and has actually already been described in section 4.5.4 as the n-dimensional zone. In terms of the VBF the virtual link is through the fundamental data type as described in section 4.5.2.

At its simplest the population data structure is a vector array of chromosomes as defined in section 4.5.5. The class defines a vector of base-class chromosome base type as described in section 4.5.4, which is derived directly from the fundamental data type as described in section 4.5.2. It is through this mechanism that full chromosome polymorphism is defined, with the proviso that any derived chromosome is derived from that virtual base-class. The population class has a slightly more complex role to play and has been extended beyond the basic vector container as it defines a high-level access mechanism to the discrete derived chromosomes.

### 4.5.6 The operators: the virtual class structure

This is defined at the base level of the VBF specification, the significant parameters of this virtual base class (**Op\_vB**) are shown in figure 4.10.

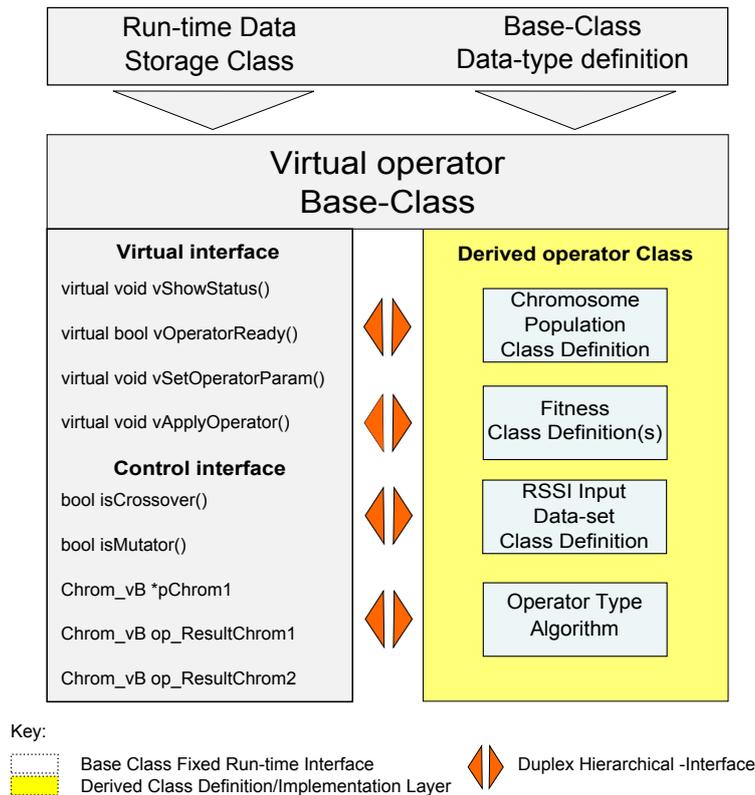


Figure 4.10: The Virtual Base-class operator block diagram

The functions prefixed as virtual form the necessary virtual interface that must be

---

implemented by the derived classes to satisfy whatever algorithm or operator specific requirements that may be relevant. Provided that the virtual functions are implemented then the derived class objects will be accessible by the VBF during the normal iterative process. The base class does not implement any form of operator functionality.

For the crossover type of operator that has been designed to operate on an array of objects (representing individual genes) simply swapping discrete array element objects, or more probably, a pointer or reference to each array element might seem all that is required.

There is a further consideration for all types of operator, the input data-set packet entries are uniquely identified by a combination of sender and receiver identification codes (and the time field). The *round-robin* system is modelled quite simply to step through each element in the chromosome, therefore to maintain the data-set integrity that binds each Drop Sonde element to its associated data, the elements objects cannot be moved within the array, (as frequently occurs in many EA implementations), as the unique identification code that defines each discrete Drop Sonde element is defined by its array index.

#### 4.5.7 The operators: The mutation operators

This operator type has a number of versions, these are described below, the derivation process for all of these operator classes is identical as they are all derived directly from the **Op\_vB** defined by the VBF.

##### Mutation group type - chromosome allele positional mutation type

1. **Move element Standard** This mutation operator provides the fundamental mutation strategy common to all mutation type operators. The essential mutation function is to modify the X, Y & Z positions for all Drop Sonde element representations in the chromosome by an amount controlled by an internally defined parameter, the actual amount of mutation is calculated using a randomly set value between zero and a pre-set maximum percentage of the absolute axis value, the latter is the `datType_B absRange` parameter as defined in section 4.5.2. This is applied by the **vApplyOperator** virtual function specified by the abstract base-class **Op\_vB** as shown in figure 4.10.
2. **Move element (Random Selection)** This mutation operator is a derivation of the **Move element Standard** operator, it also Randomly modifies the X, Y & Z positions for a number of Drop Sonde elements. These elements are randomly selected from any position in the chromosome up to a pre-defined limit.
3. **Move element (Window)** This mutation operator is another derivation of the **Move element Standard** operator, but it shares a similar operational mode to

## 4. SOFTWARE MODEL DESIGN

---

the **Move element (Random)** operator. The difference being that the set of motes are selected as a contiguous group from a randomly selected index start point, the group may be restricted if the number of elements to be selected is greater than the remaining number in the chromosome from the start index.

### Mutation group type - chromosome indexed-pair swap

1. **Swap section** This operator does not actually mutate the X, Y & Z positions directly, it swaps the X, Y & Z values between a pair of randomly selected Drop Sonde elements as represented by discrete alleles from the same chromosome. In this way it is similar in operation to the **Move element (random)** operator described above except that there is no new positional data generated. The effect of the operator is analogous to a positional jump rather than a small controlled mutation.
2. **Shunt section** This operator is similar in operation to the **Swap section** operator except that it swaps the X, Y & Z values between one defined chromosome contiguous section and another of same size, these may overlap. The effect of the operator is also analogous to a positional jump rather than a small controlled mutation.
3. **Invert section** This operator functions in a similar way to the **Swap section** operator, it takes a contiguous chromosome section of more than one element and reverses, or inverts, the X, Y & Z axes values of the section; specifically the X, Y & Z axes of selected section element one are swapped with X, Y & Z axes from the selected section element n-1, and so on until all pairs are swapped. This is also analogous to a positional jump rather than a small controlled mutation.

### 4.5.8 The operators: the crossover operators

This operator type has a number of versions, these are described below, the derivation process for all of these operator classes is identical as they are all derived directly from the **Op\_vB** defined by the VBF.

#### 1. The single point crossover

This operator merges two discrete *parent* chromosomes that have been *cut* at a single, randomly selected, index cut-point. The new *child* chromosome is built by copying all element X, Y & Z data from the first parent chromosome indices zero to (cut-point - 1), the second part of the *child* is built from the second *parent* chromosome from the cut-point to the last index position. This is illustrated in table 4.2.

Single point Crossover															
Chromosome_1	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Chromosome_2	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
Index point							↓								
Child_1	A	B	C	D	E	F	g	h	i	j	k	l	m	n	o
Child_2	a	b	c	d	e	f	G	H	I	J	K	L	M	N	O

Table 4.2: Single point operator

## 2. The n-point & uniform crossover

This operator also merges two discrete *parent* chromosomes, the difference is that there are a randomly selected number of index cut-points, up to a maximum of the length of the current chromosome. The new *child* is built by copying all element X, Y & Z data from the first parent chromosome indices zero to the first (cut-point - 1), the next part of the *child* is built from the second *parent* chromosome from the current cut-point to the next (index cut-point - 1) index position. An example of the n-point approach is illustrated in table 4.3.

n-point Crossover															
Chromosome_1	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Chromosome_2	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
Index points							↓				↓				↓
Child_1	A	B	C	D	E	F	g	h	i	J	K	L	m	n	o
Child_2	a	b	c	d	e	f	G	H	I	j	k	l	M	N	O

Table 4.3: n-point Crossover operator

Repeatedly increasing the number of index cut-points will result in operator behaviour that increasingly resembles that of the uniform crossover method, until full uniform ‘*operation*’ is reached when the number of index cut-points equals that of the chromosome length as shown in table 4.4.

Uniform Crossover															
Chromosome_1	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Chromosome_2	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
Index points	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
Child_1	A	b	C	d	E	f	G	h	I	j	K	l	M	n	O
Child_2	a	B	c	D	e	F	g	H	i	J	k	L	m	N	o

Table 4.4: Uniform Crossover operator

## 4. SOFTWARE MODEL DESIGN

### 4.5.9 The objective fitness: 3-dimensional positional error class

In this section the intention is to describe the general run-time operation of the first level derived class structure objective fitness function. The operational mode of the geometric reconstruction model, at its most fundamental, is through the modification, or mutation, of the 3-dimensional positions of all, or a sub-set, of the Drop Sonde elements represented in each chromosome.

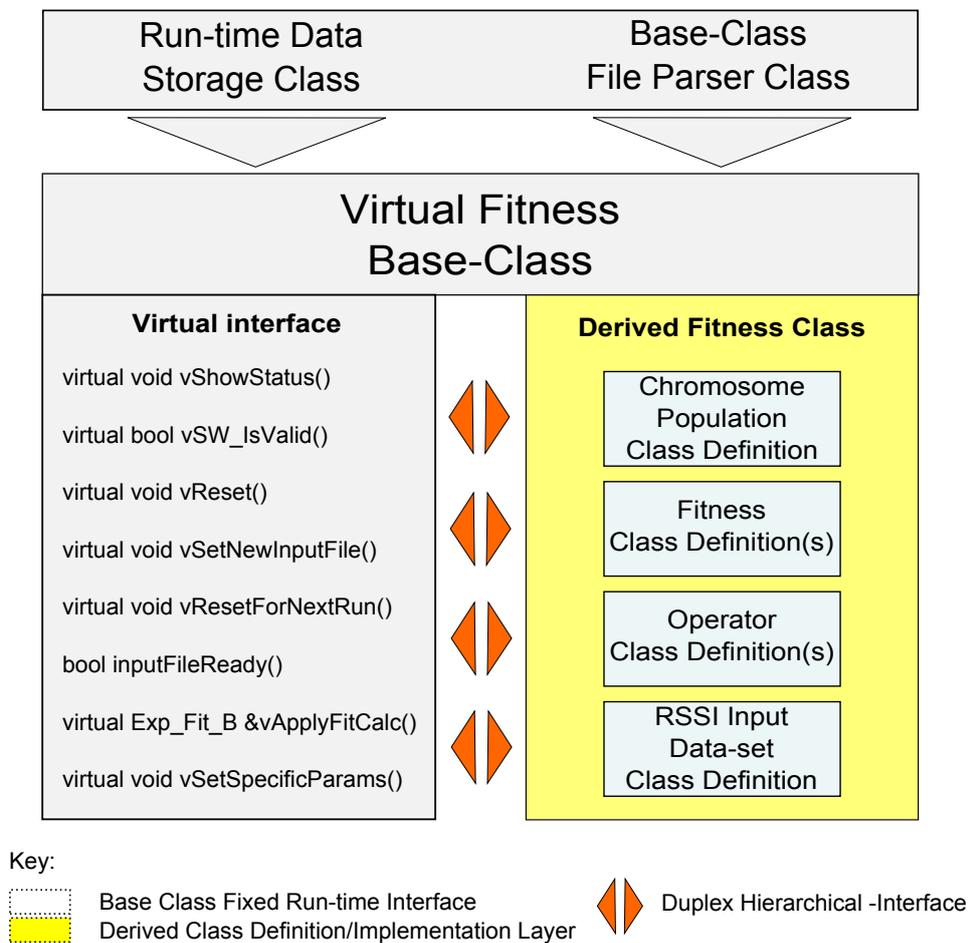


Figure 4.11: The derived fitness block diagram

After every iteration the active objective fitness function is applied to each chromosome in the population in turn, the results stored during the model run for subsequent post-processing. In order for the correct run-time integration in the software model the object class containing the fitness functions used for all simulations in this thesis must be derived from the exp.Fit\_B virtual base-class defined as a fundamental part of the

---

VBF interface; the overview of this interface definition is shown in figure 4.11.

This interface defines a set of virtual functions that must be supplied, (ie. coded), in the derived class methods to perform the single, multiple, hybrid, or heuristic, objective fitness calculation as defined by the requirements inherent in the problem space definition. This is a first level VBF derived class structure that can also be used to provide the basis for additional class derivations that may be required for more complex simulations where the objective fitness calculations define specific variations to comply with simulation operator and algorithm requirements.

A simplified definition of the generic single-objective fitness function is shown in figure 4.12, this defines a generalised approach to the fitness calculation as a sub-set of the overall set of real numbers; these sub-sets also include real number representations of the set of positive and negative integers.

Where:

Function  $f : S \rightarrow R$  from set  $S(R)$  of real numbers (including integers)

$S_{max}$	The Set of real numbers larger than $f(x)$
$S_{target}$	The Set of (one) real number equal to $f(x)$
$S_{min}$	The Set of real numbers smaller than $f(x)$

For a generic **maximising** function

Find an element  $x_n$  in S such that  $f(x_n) \geq f(x) \forall x$  in  $S_{max}$

For a generic **target** function

Find an element  $x_n$  in S such that  $f(x_n) = f(x) \forall x = S_{target}$

For a generic **minimising** function

Find an element  $x_n$  in S such that  $f(x_n) \leq f(x) \forall x$  in  $S_{min}$

Figure 4.12: Fundamental fitness function types

These generic fitness definitions do not contain any definitive information as to how the value representing element  $f(x_n)$ , or more precisely chromosome  $f(x_n)$ , is to be determined from the overall base set of real numbers  $S(R)$ . The determination of this data will, in nearly all cases, be subject to a process of calculation derived from a complex single or multiple-objective fitness function, the operation of which is defined by the problem space context.

## 4. SOFTWARE MODEL DESIGN

---

### 4.5.9.1 The default objective fitness calculation

The objective fitness function for all simulations in this thesis are based on the definition of a *tending-to-zero minimising* function mode as described in figure 4.12. This definition is used in a modified form as shown in figure 4.13. A significant difference in this definition is that the minimum, (the ideal fitness in this instance), is limited to zero as it is not arithmetically possible to get a negative result; to do so would represent an error as the relative positional offset for all elements in the model are represented as magnitudes which are not signed.

Where:

Function  $f : S \rightarrow R$  from set  $S(R)$  of real numbers (including integers)

$S_{min}$  The Set of real numbers smaller than  $f(x)$

For a generic *tending-to-zero minimising* function

Find an element  $x_n$  in  $S$  such that  $(0.0 \leq f(x_n) \leq f(x)) \forall x$  in  $S_{min}$

Figure 4.13: Default objective fitness function

Each Radio Sonde element is represented by a unique numeric identifier in the range (zero to element cluster size), this is also used as the allele index that the element occupies in the chromosome, (the gateway is always at index zero); this identifier, (and the equivalent allele index) remain unchanged throughout the run for all chromosome members of the population. This is to ensure that when a pair of transmitter (source element) and receiver (remote element) indices are selected during the objective fitness function process the elements they represent will always be consistent between the chromosome position they occupy and the input data-set representation.

The ideal or perfect objection fitness represented by the *minimised* 3-dimensional relative positional error value can be informally defined as shown in figure 4.14.

*“The objective fitness is derived from the total squared sum of all inter-element distance magnitude differences calculated from the modelled 3-dimensional Cartesian positions and the corresponding data derived from the RSSI input data-set. The ideal, or perfect, objective fitness of zero positional magnitude error being a direct representation of the correct 3-dimensional positions of all elements relative to all others in the cluster”.*

Figure 4.14: Informal objective fitness definition

---

#### 4.5.9.2 Single element pair distance magnitude difference calculation

The default objective fitness function calculates the total objective fitness for a complete chromosome from the sum of all positional magnitude difference values between all possible combinations of discrete pairs of elements. The distance magnitude for pairs with the same source and remote ids are not calculated, (the result is always zero), however this data is stored to keep the number of calculations consistent with the number of elements.

Figure 4.15 shows a simplified cluster that, for the purposes of illustration, consists of a single pair of elements. In this example the transmitting and receiving elements, (shown at  $Element_{rem}$  &  $Element_{src}$  respectively), are in a relationship where the transmitting element is fixed at the notional origin, (the gateway), whereas the receiver element is able to move freely.

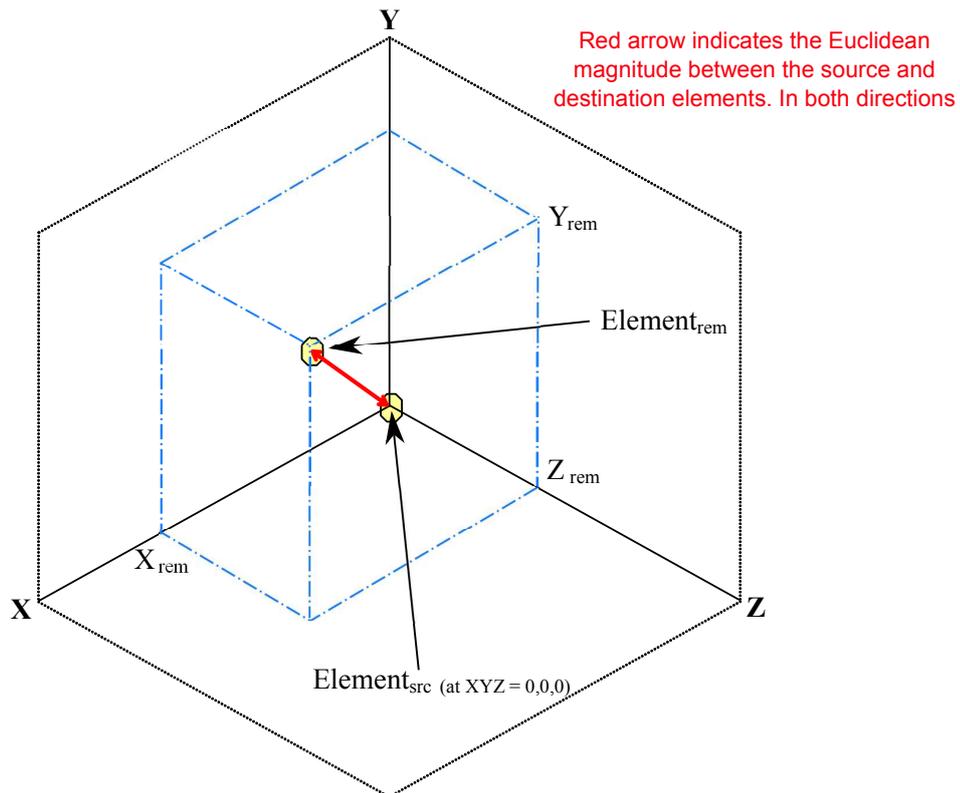


Figure 4.15: Euclidean magnitude between discrete element pair

To calculate the magnitude difference error between the modelled source/remote pair and the equivalent pair from the input data-set, both values must be in a form that

#### 4. SOFTWARE MODEL DESIGN

---

is directly comparable. The input data-set data contains only non-vectorized magnitude values which can have no coordinate information, nor can this be readily determined, (this is after all the reason that the software model exists).

The first step is to convert the chromosome representation of the 3-dimensional Cartesian point location for a pair of elements into a simple non-vectorized magnitude between discrete pairs of elements using the 3-dimensional analogue of Pythagoras' theorem ( $\sqrt{X^2 + Y^2 + Z^2}$ ) to give the 3-dimensional magnitude distance value represented by the 3-dimensional *'hypotenuse'*.

The modelled part of 3-dimensional magnitude distance value pairs can now be directly compared with that of the input data-set indexed non-vectorized magnitude and an error value determined that reflects the difference in modelled distance and actual (RSSI derived) magnitude distance. When the magnitude of these values are identical the relative positional error (for this pair of elements) will be zero. In this ideal case the original Cartesian coordinates for the discrete pair of elements are now also valid, but only in the relative *'localised'* context defined by those specific elements. Although part of the reconstructed 3-dimensional geometry has now been determined the validity is only for this iteration as these are likely to be modified as part of the active optimisation algorithm.

Equation 4.16 shows the simple magnitude calculation between a source and remote element pair based on the Cartesian axes representation used within the model when calculating the error between the current 3-dimensional positions of a pair of elements and that *'encoded'* in the RSSI based input data-set. This represents the non-vectorized Euclidean magnitude of the 3-dimensional *'diagonal'* distance between the two points of the notional cuboid depicted with the blue outline in figure 4.15; this is the simplest possible case as the one of the elements is at the origin with coordinates of 0, 0, 0.

To put this into the context of the operation of the *'Drop Sonde'* device, the RSSI input data-set consists of a set of RSSI derived magnitude values without any form of definitive coordinate information, hence the need for a modelling system to represent the 3-dimensional geometry, in this instance a Cartesian based reference frame is employed.

---

Where:

Src: The source cluster element

Rem: The remote cluster element

Dist: 3-dimensional distance between Source & Remote Cartesian points

$$|Dist| = \sqrt{(|X_{Src} - X_{Rem}|)^2 + (|Y_{Src} - Y_{Rem}|)^2 + (|Z_{Src} - Z_{Rem}|)^2} \quad (4.1)$$

Figure 4.16: Single element Cartesian distance magnitude

Equation 4.16 shows the ‘one-way’ calculation between a single pair of elements, the result is the magnitude of the distance between two discrete 3-dimensional points; it is this magnitude value that is compared to the RSSI derived distance magnitude value from the input data-set between the same source/remote pair of elements. The magnitude difference between the modelled 3-dimensional Cartesian position and that derived from the RSSI input data-set indicates the error in the relative positions of the two elements for the two states (ie. the modelled and that of the input data-set), these values are then squared and then summed to create a ‘sum-of-squares’ final value; when this equates to zero then these two elements are in the correct relative locations to each other, this is part of the definition of ‘local fitness’.

This equation shows only this single pair calculation, to complete the set of values pertinent to this pair of elements it must also be performed with the source/remote pairing reversed as the RSSI input data-set magnitude values are likely to differ for an apparently identical distance magnitude; in the complete fitness function calculation this is extended for all elements pairs using the ‘round-robin’ method described in section 4.2.1.

#### 4.5.9.3 Discrete element intermediate local objective fitness

The ‘local fitness’ of a single discrete element is simply one where the 3-dimensional error pertinent to the 3-dimensional position of a single, discrete element relative to all others in the cluster is calculated in isolation. This step directly reflects how the objective fitness is actually calculated, with all local fitness values for all elements combined into the overall fitness, this is a natural step imposed by the ‘round-robin’ method described in section 4.2.1; in summary, this method requires that each element transmits in isolation and all others must be in ‘receive’ mode to avoid the corruption to the RSSI value caused by transmission clashes.

This methodical approach results in a sub-set of the complete RSSI input data-set that is restricted to RSSI positional data specific to the position of transmitting ele-

#### 4. SOFTWARE MODEL DESIGN

---

ment and relative to the set of receiving elements. This *‘intermediate’* result can also be stored in the element object and used to provide a sequential *‘history’* of how the objective fitness for each element progressed throughout the simulation run. Taken in isolation this information is useful in the sense that it gives an indication of the local objective fitness of an individual element relative to all others in the cluster. As the complete RSSI input data-set is strongly inter-dependent with respect to all of the elements represented in the cluster the scope of applicability of the local fitness data is limited, but can be used to indicate which of the elements in the cluster is *‘most out of position’* relative to all others.

Any positional changes to the local element will, by default, have an effect on the overall objective fitness of the entire cluster, (as represented by the chromosome), due to the inter-dependency implicit in the input-data-set *‘encoded’* geometry. The normal selection process of the EA will ensure that any resultant reduction in objective fitness will, ideally, be *‘evolved away’*, whilst simultaneously, adding to the diversity in the chromosome population. The information in this intermediate form is not currently used in any objective appraisal of the run-time performance of the reconstruction model, but it does have the potential to provide useful data for both a heuristic operator and in an overall *‘quality factor’* (in conjunction with the Range Group type of method in section 4.6.5) in some future work.

---

#### 4.5.9.4 All element error summation calculation

The complete objective fitness calculation process is the full combination of all discrete intermediate element local objective fitness values, the pseudo-code in figure 4.17 illustrates the high-level operation of objective fitness function.

```
START
  SET: Summed_value to zero
  SET: Total_element_count to zero
  LOOP: FOR ALL cluster elements (0..n)
    LOOP: FOR ALL cluster elements (0..m)
      SET: Intermediate_fitness_value to Zero
      IF: ( $n \neq m$ )
        CALCULATE: Distance_magnitude difference between elements(n, m)
        SQUARE: Distance_magnitude value
        INCREMENT: Intermediate_fitness_value by Distance_magnitude
      END IF:
      INCREMENT: Total_element_count
    END LOOP: (0..n)
  INCREMENT: Summed_value by Intermediate_fitness_value
END LOOP: (0..m)
END
```

Figure 4.17: Pseudo-code of objective fitness function

Figure 4.18 depicts the ‘RSSI connections’ between the cluster elements ( $M1$ ,  $M2$  &  $M3$ ) and the gateway element  $G$ , there are two connections for each element pair as during the ‘round-robin’ stage each discrete element first acts as sole transmitter and then acts as receiver for the transmissions from all other elements; the RSSI data is uniquely identified by this source/receiver ‘ident’ pair data.

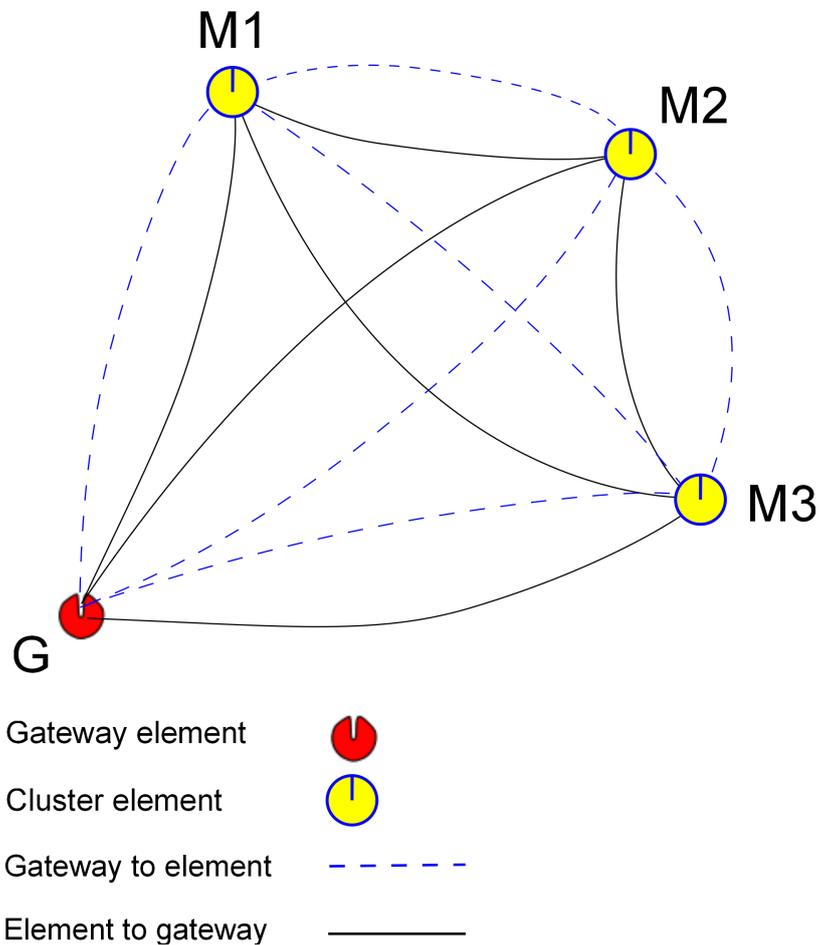


Figure 4.18: Discrete inter-element RSSI magnitude data points

The complete set of inter-element pairs generated for this simple element cluster, in both directions, is shown in figure 4.19, this is in the form of transmitter/sender and represent the RSSI magnitude data from the input data-set that the receiver elements have generated from the transmitter element during the ‘round-robin’ phase.

<b>G to M1</b>	<b>M1 to G</b>	<b>M2 to G</b>	<b>M3 to G</b>
<b>G to M2</b>	<b>M1 to M2</b>	<b>M2 to M1</b>	<b>M3 to M1</b>
<b>G to M3</b>	<b>M1 to M3</b>	<b>M3 to M2</b>	<b>M3 to M2</b>

Figure 4.19: Data-set inter-element identifier pairs

---

Where:

k	The number of elements in the cluster
$S_{rssi}$	Indexed input RSSI dataset
$d_{xyz}$	Euclidean magnitude distance between XYZ points n.m
$d_{rssi}$	Euclidean distance from input dataset
src	source identifier/chromosome allele index/RSSI input data-set source index
rem	chromosome allele index/RSSI input data-set remote index

$$Fit = \sum_{n=0}^{n=k} \sum_{m=0}^{m=k} n \neq m \rightarrow ((d_{xyz} - d_{rssi})^2 : d_{rssi}[n, m] \in S_{rssi}[n, m]) \quad (4.2)$$

Figure 4.20: All element objective fitness function

The objective fitness function for all elements pairs is shown in figure 4.20, this simple equation represents the summation of the magnitude difference errors between all pairs of cluster elements in both directions for a single iteration.

The complete iterative results consists of a set of discrete objective fitness values for individual runs automatically stored in the VBF defined ‘*DVec*’ data collation modules described in section 4.6.2; these data collation modules supply all data for the objective fitness data plots.

## 4.6 The data storage, collation and output sub-system

A significant section of the Virtual base-class Framework (VBF) involves the Data Storage Sub-system (DSS), this is a fully implemented set of classes that provide the capability for run-time data storage, simple statistics generation and formatted file output. The suite of classes have a large, but fairly simple, user interface that has been designed to conceal the complexity inherent in the discrete and compound data handling and fundamental post-processing required by the simulation.

The DSS class framework is embedded in the VBF framework at a fundamental level through the ZoneFitDV wrapper class. This class provides the interface to a defined set of encapsulates data storage and manipulation objects. This object has the specific task of coordinating instantaneous data storage, post-processing and results file output across many discrete runs.

## 4. SOFTWARE MODEL DESIGN

---

### 4.6.1 The DSS elemental class types

At the fundamental level the DSS class hierarchy consists of two direct access data storage primary base classes and a set of secondary classes that each implement an interface consisting of the set of discrete and defined functions essential for various aspects of data file formatting and data manipulation.

Class name	Description	Level.
<b>ZoneFitDV</b>	Data access interface VBF parent class	VBF
<b>DVec</b>	Vector of double precision real numbers	Primary
<b>DVecV</b>	Vector of DVec objects	Primary
<b>DatStat</b>	Single line statistics formatting class	Secondary
<b>DatStatV</b>	Vector of DatStat objects	Secondary
<b>RangeGroup</b>	Single line group classifier class	Secondary
<b>RangeGroupV</b>	Vector of RangeGroup objects	Secondary
<b>DVec_FCont</b>	Input/output file/format control class	Secondary
<b>ArrayFileData</b>	File creation/verification helper class	Secondary

The Dvec(V) classes have been designed to enable instantiation as stand-alone objects, the secondary classes, as the label suggest are required only when their specific functionality demands their inclusion through run-time binding.

The complexity of the DSS combined class operation precludes anything approaching a complete description, therefore the discussion will be limited to some of the more useful basic functions used and the post-processed data output generated as a result. The VBF inter-connection diagram is depicted in simplified form in the block diagram in figure 4.21.

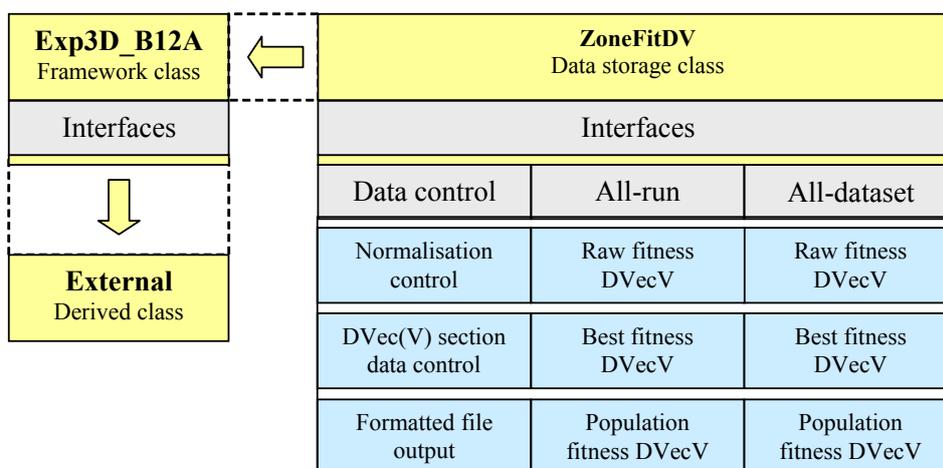


Figure 4.21: Embedded VBF automated data storage class block diagram

This class is fairly simple in structure, it has the primary purpose of providing a set of default DVecV classes from within the same object along with a simple set of access and control functions that, to a significant extent, mirror those of the DVec(V) classes introduced below.

This interface enables the common functions that can be applied across DVecV objects as single entities or in simultaneously in a group when required. The run-time data access and output functionality provided by the ZoneFitDV class interface provides the interface for the standard, or default, data collation and output functionality controlled by the logic defined within the iterative control class.

#### 4.6.2 The DVec storage object

The DVec object is the principal data collation object within the DSS interface, this object provides the fundamental functionality required for the run-time discrete double precision real number storage and statistics generation sub-system for any simulation derived from the VBF. The DVec object is built using the standard vector base-class and several associated secondary classes that each define specific functionality that provides specific functionality additional to the fundamental purpose of simple data storage.

The default usage mode, as implemented in the VBF iterative control class, results in an object of this class being instantiated for each discrete run in a simulation, this provides the run-time storage of the raw fitness data as it is generated. On the completion of the run the current DVec object is transferred to the relevant DVecV contained in the ZoneFitDV embedded wrapper object for storage until the end of the simulation. The overall interface structure is shown in figure 4.22.

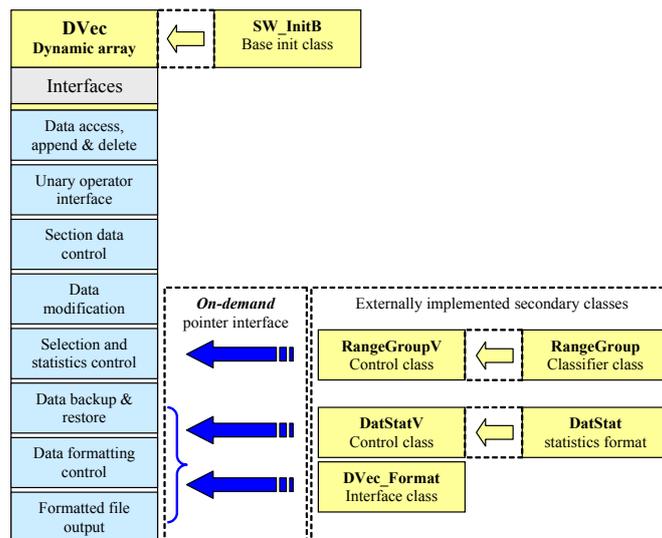


Figure 4.22: DVec dynamic array class hierarchical block diagram

## 4. SOFTWARE MODEL DESIGN

---

The overall functional interface for this object has been notionally sub-divided into a number of area of defined DVec access and control for easier functional description; all defined functionality are available at all times through the object pointer or reference. The most commonly used notional interfaces are briefly summarised in subsequent sections.

### 4.6.2.1 The DVec access and modification interface

These functions for the primary interface that most applications would use in general run-time processing, most common here are the append group of functions. The list below is a very basic summary of some of these functions, the function types listed below are shown in their default form, in addition to these they all have a number of overloaded counterparts that have parameters appropriate to the function such as those defining start/end indices.

Function name	Description
appendVal()	Insert new data at the end of the vector
prependVal()	Insert new data at the start of the vector
insertVal()	Insert new data in the vector at index
replaceAt()	Replace data at index
replaceVals()	Replace data that matches criteria
remValsAt()	Remove data at index
remNegVals()	Remove all negative data
remPosVals()	Remove all positive data
modSqu()	Square all vector data
modRcp()	Apply reciprocal to vector data
modPos()	Make all vector data absolute
modNeg()	Make all vector data negative
modTog()	Toggle sign of all vector data
modCeil()	Promote all vector data to absolute integer
modFloor()	Demote all vector data to absolute integer

### 4.6.2.2 The DVec meta-data and statistics interface

In addition to the access and modification type of interface the DVec class implementation also implements a set of meta-data functions that return a specific value calculated directly from the vector of stored values. This can take the form of a simple selection function (ie. getMax), or take the form that will calculate the result of a simple statistical function (ie. getVariance).

The list below is a summary of these functions, as with the access and modification functions these types listed below are also shown in their default form, in addition to these they all have a number of overloaded counterparts with additional parameters appropriate to the function.

---

<b>Function name</b>	<b>Description</b>
<code>getAvg()</code>	Average of vector data
<code>getMax()</code>	Maximum of vector data
<code>getMin()</code>	Minimum of vector data
<code>getSum()</code>	Sum of vector data
<code>getDiff()</code>	Difference between specified indices
<code>getGradient()</code>	Gradient between specified indices
<code>getMedian()</code>	Median of vector data
<code>getStdDev()</code>	Standard deviation of vector data
<code>getVariance()</code>	Variance of vector data
<code>getRangeGroupings()</code>	Range group classification of vector data

Of these, the average, minimum and standard deviation functions are used most extensively in the generation of the meta-data that forms the majority of the final compound-run results data.

The range group classification meta-data interface is accessed via the embedded `DVec` class access pointer to the externally defined and instantiated `RangeGroup(V)` objects that implement range group classification functionality. This object provides a significant and fundamental data type, that is used both directly, and as the basis for a quantification metric throughout the thesis; this class is described separately in section [4.6.5](#).

### 4.6.3 The `DVecV` dynamic `DVec` array storage object

This class encapsulates a dynamic array of `DVec` objects thereby providing the mechanism to store individual discrete run data objects automatically when running a compound-run simulation, and to provide additional statistical operations on those objects when considered as a whole set of discrete runs from complete simulations.

The overall interface structure is shown in figure [4.23](#).

## 4. SOFTWARE MODEL DESIGN

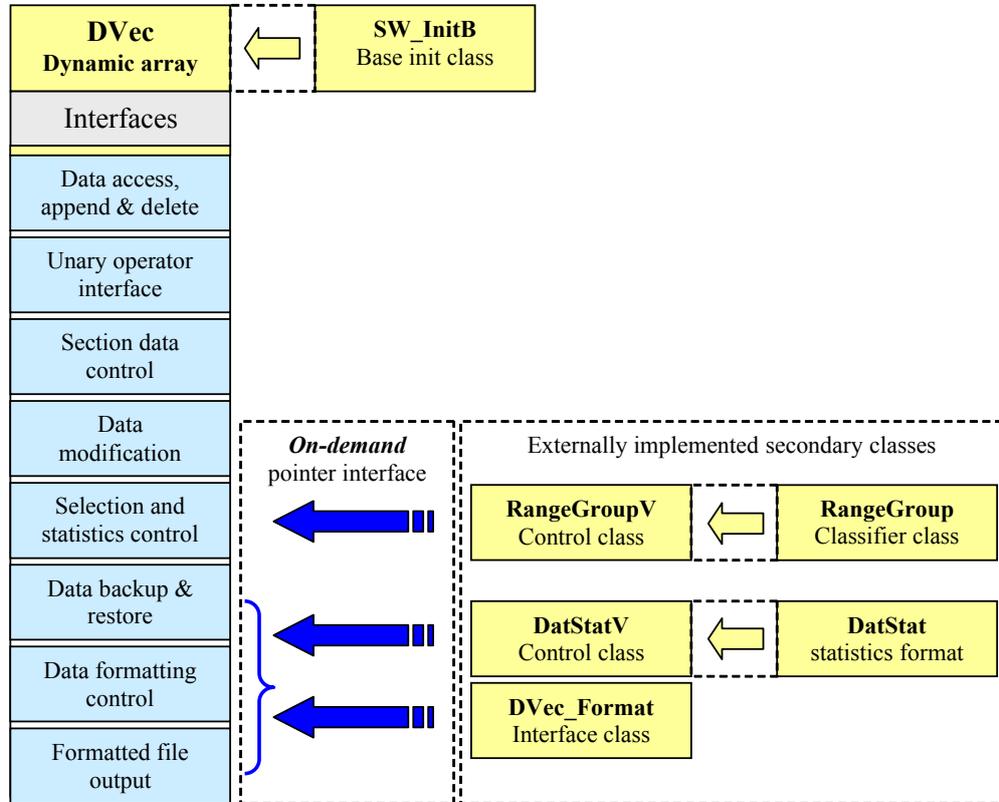


Figure 4.23: DVecV dynamic array class hierarchical block diagram

### 4.6.4 The DVecV Meta-data and statistics interface

The statistical meta-data is generated by the DVecV class using the set of meta-data functions defined in the DVec class, the DVecV class is itself derived from the DVec class, and as a result, also implements these same meta-data functions. The statistics can be generated *‘vertically’* for each discrete run, (using the functionality provided by the DVec class), and *‘horizontally’*. In the latter case the objective fitness data across all runs at any given iteration can be considered as a separate data storage entity and the statistics applied to them, this is illustrated in appendix E.3, the associated DVec sourced data columns is shown in the previous section in appendix E.1.

### 4.6.5 Range-group classifier meta-data classes

The range classifier classes provide a two-tiered system by which an array of numerical data can be classified as being within a pre-set range that has been defined by a text string representation of a simple upper and lower limit and associated mathematical operators. This type of meta-data is quite applicable at single run DVec level, in the context of this problem space however, it is more useful that the group classification is applied across all runs simultaneously.

---

The first tier (the `RangeGroup`) encapsulates a complete, self-contained single classification range group function. The second tier (the `RangeGroupV`) encapsulates a vector of the `RangeGroup` objects to automate the instantiation of a multi-group range classification following the application of discrete or arrayed numerical data. In the context of this problem space the `RangeGroupV` classification functionality has been used to provide an alternative representation of the overall fitness. This mode detects significant step changes in the distribution of fitness values range-group classifications over an entire simulation run. The `RangeGroup(V)` classifier is directly accessed through an embedded `DVec(V)` class access pointer.

#### 4.6.5.1 The `RangeGroup` single range classifier class

This class has been designed to operate as a stand-alone object, it provides the complete implementation of a single, self contained range-group definition. The object applies a Boolean decision on whether the current externally applied numerical data is within a minimum/maximum range, this data can be a discrete value or supplied as an array. The results of a single or multiple value decision operation are available in a range of text and numeric forms, the forms used here are a simple text string representation and the number of data values that are within a specific group, in the terminology of the thesis this will be latterly is referred to as the *hits metric*.

The initialisation interface to the `RangeGroup` object is through a simple text string that defines a range that a value must be within before it is assigned to a range group. The group range definition text string follows the simple format as shown in Figure 4.24.

Where:

:	The position holder for the data value to be classified
Range left	The left hand range limit (blank indicates +/- data type limit)
Range right	The right hand range limit (blank indicates +/- data type limit)
opA	The operator for the left hand range
opB	The operator for the upper hand range
;	Range group terminator

The set of permitted operators :  $<$ ,  $\leq$ ,  $=$ ,  $\geq$ ,  $>$

**Range group** =: **Limit left** **opA** : **opB** **Limit right** ;

Figure 4.24: Single range-group definition string

The class parses the text string for the correct format before converting the text rep-

#### 4. SOFTWARE MODEL DESIGN

---

resentation into a set of variables that control a logical selection function to determine whether the candidate value is within the defined range.

Example =: ( 0.0 <= : < 5.0 ; )

The simple example above is translated as; to be classified in this range-group the candidate data value must be greater-than-or-equal-to zero AND less-than five.

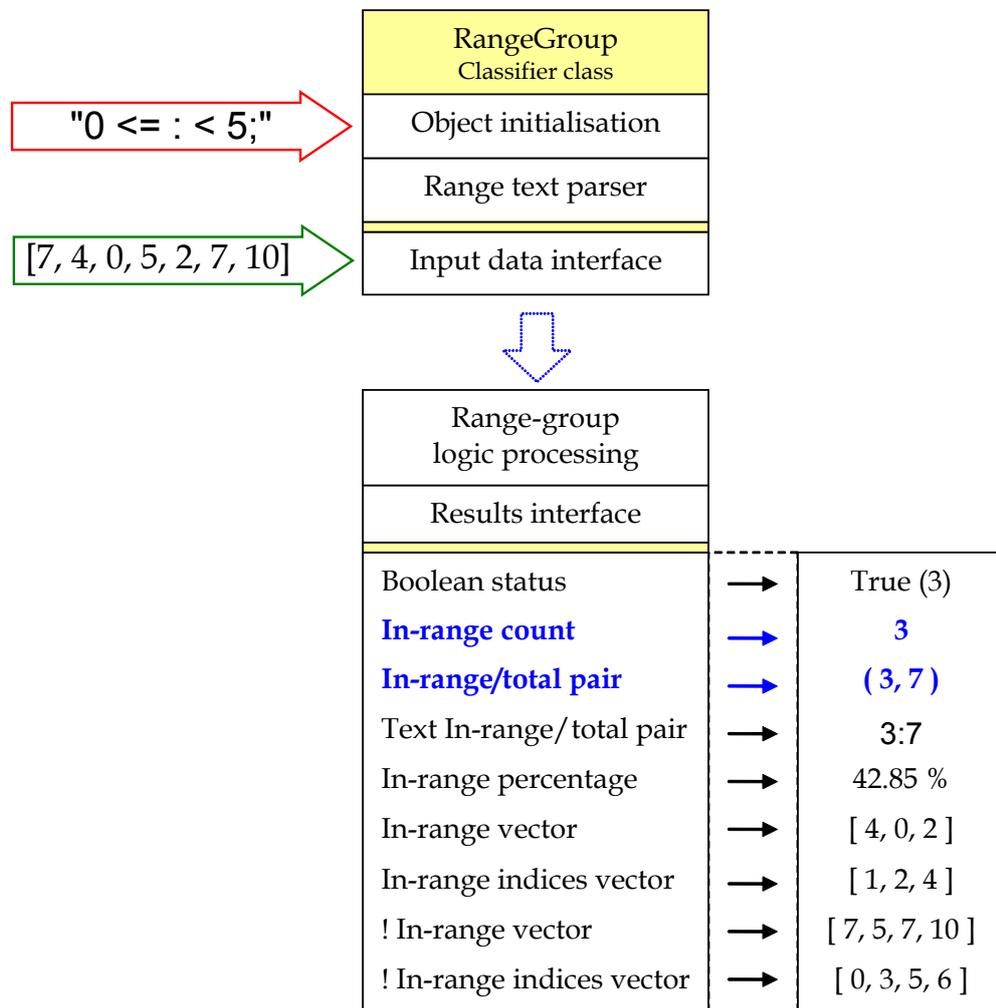


Figure 4.25: Discrete range-group function diagram

The range-group classification can also be negated to indicate the candidate data value is NOT in this range-group for all values less-than zero OR greater-than-or-equal-to five. For a single candidate data value then the result is a simple Boolean, when mul-

tuple values have been classified, singly through the discrete data function, or via the overloaded real number array function, the classification results are now available in other forms, these are summarised in the *Results interface* shown in figure 4.25.

The object instantiation and initialisation requires a group range definition text string for group classification data (the red arrow), this can be modified at any time during the operation of the object. The green arrow indicates the numeric data to be classified by this group, in this instance it is an array of real number values. The brown dashed arrow represents an optional scaling factor that can be applied to the count of in-range values, this is useful for applying some externally defined *weighting scheme* for example; this is also the purpose for which this is used in the simulation results and appraisals throughout the thesis.

The results types and results values appropriate to the actual input definitions of the red and green arrows are summarised in the *Results interface* block; the *In-range count* & *In-range/total pair* (shown in blue) are the aspects of this interface used to create the basic run-time results classification by the RangeGroupV object described below.

#### 4.6.5.2 The RangeGroupV multiple range classifier class

This object has the primary task of converting a compound range-group text string into a vector of RangeGroup objects, each vector entry that conforms to one of the text string segments of the discrete type as defined in figure 4.26. The result of this is a series of range groups, the output of which can be concatenated to provide a cascading range classifier that returns a compound data structure, or text equivalent, defining the range classification data for the applied real number array sourced from the relevant DVec data object.

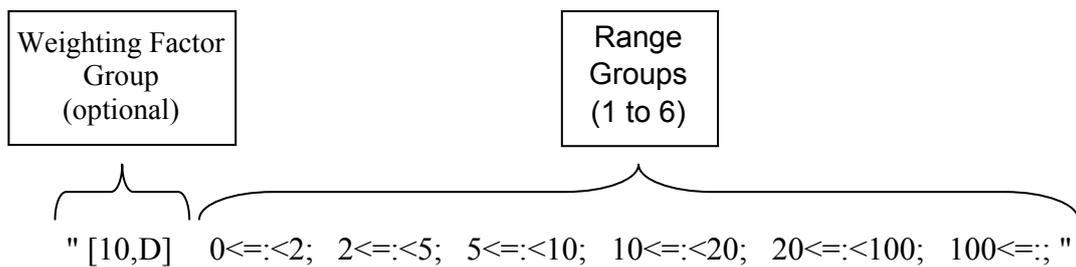


Figure 4.26: Example RangeGroupV input text string

The compound range-group text string, shown above, is taken directly from the initialisation file, with the exception of the optional *Weighting Factor* group as the default values are appropriate for the requirements of this work, this function is described separately in the next section. This line details the contiguous range groups used for

## 4. SOFTWARE MODEL DESIGN

---

all simulations in the thesis. The lower and upper limits for the individual groups have been set through empirical observation of the general run-time performance when modelling the 3-dimensional reconstruction problem; they have been found to offer a reasonable indication of model simulation performance on many variants of the software modelling. In this example all groups are contiguous, groups 1-5 cover the entire range of expected fitness values, whereas group six has its upper limit set to infinity to ensure that even the most unfit results are classified.

### 4.6.5.3 The RangeGroupV: weighting factor group

The final part of the RangeGroupV class definition concerns the *Weighting Factor* group, this is an optional part of the input text string as shown in figure 4.26. The purpose of this group is to provide a set of weighting factors that can be applied as scalars to each discrete range group within a RangeGroupV defined set. The general format of the text instruction is shown in figure 4.27.

Where:

```
(    Limit left
)    Limit right
A    Factor direction: Ascending left to right (sequential multiplication)
D    Factor direction: Descending left to right (sequential division)
;    Weighting group terminator
```

**Weighting Factor** =: ( **Initial Value** , **Factor direction** ) ;

Figure 4.27: Optional Weighting Factor group definition string

The output of this function is a sum of the weighted *hits* per RangeGroup as shown in equation 4.3, the number of *hits* within each of the RangeGroups is multiplied by the relevant weighting factor for that RangeGroup and then summed.

Where:

m            The number of Range Groups  
 $S_{RG}$         Set of indexed weighting factors  
 $S_{Hits}$       Set of indexed input Range group *hits*

$$Weighted\ Hits\ Metric = 1.0 / \sum_{n < m}^{n=0} S_{RG}[n] * S_{Hits}[n] \quad (4.3)$$

The function produces a group of weighting factors, one for each RangeGroup specified in a RangeGroupV, the **initial value** is used as the starting point and assigned to the

---

first (leftmost) RangeGroup. The next RangeGroup receives the weighting factor for the previous RangeGroup modified according to the **Factor direction**, that is, multiplied or divided by ten, this continues for all RangeGroups. This module is currently configured with default settings to apply the weighting factor for all RangeGroups with a sequential ascending or descending step of ten. Using the example RangeGroupV text string with the optional weighting factor string **(10,D)** as defined in figure 4.26, this results in the set of weighting factors as shown in figure 4.5.

Range Group						
Type	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
Format	0<=:<2	2<=:<5	5<=:<10	10<=:<20	20<=:<100	100<=:<
Weighting	10.0	1.0	0.1	0.01	0.001	0.0001

Table 4.5: Range Groups and Weighting Factors

#### 4.6.5.4 The RangeGroupV: weighting factor formatted output

using the set of RangeGroups and associated Weighting factors shown in table 4.5 and the same set of input data as used in figure 4.26. In the context of the work of the thesis the set of input data reflects the multiple-run mode of generating averaged results.

Input data → [7, 4, 0, 5, 2, 7, 10]

$$WeightingFactor = 1.0/(1*10) + (2*1.0) + (3*0.1) + (1*0.01) + (0*0.001) + (0*0.0001)$$

Output string → "0.8123476 |1 |2 |3 |1 |0 |0 |(7 7)"

An additional format for this data is the *Integration value* that is used in many of the tabulated results tables. This is, as the name implies, a method of integration applied to the iterative, instantaneous, *Weighting Factor* group values (as described above). In effect the *Weighting Factor* is calculated for each iteration and stored in the output file as discrete values, these can be then summed to provide the exact *area-under-the-graph* as the simulation run progresses. In this implementation, where the active fitness function is minimising, the closer this value is to zero then quicker the convergence is occurring. Although this is an empirical measure it has been very useful in assessing the overall performance of comparative simulations.

#### 4. SOFTWARE MODEL DESIGN

---

## Part III

# *'Standard'* Algorithm Simulations



## Chapter 5

# Software model performance base-line using non-optimised algorithms

The main objective for this chapter is to present an overview of the process of implementing a simple, Non-optimised, software model based on the set of Standard Derived Classes (SDC) directly derived from the Virtual Base-class Framework (VBF) as described in chapter 4. The *SDC* refers to the set of C++ classes that form the basis of the run-time software, the run-time definition of these classes is algorithm specific; notable differences are highlighted in the relevant chapter or section.

When considered together, the elements defined by the combined VBF and SDC entities implement the hierarchical, functional, and syntactically correct, requirements of a software model designed to fulfil the specific 3-dimensional geometric reconstruction optimisation task. To complete a run-time reconstruction model that is semantically meaningful requires at least one algorithm to be implemented that has been designed to investigate specific aspects of the problem space.

The first part of the chapter investigates the inherent ability of the software model to solve the 3-dimensional reconstruction problem following a purely random approach to the generation of candidate solutions. A pair of externally derived, pseudo-random, mutative, algorithms are defined; these are derived directly from the SDC implementation of the chromosome format specified for this problem space. These algorithms are intended to be intrinsically non-optimal as the search for candidate solutions will be subject to an absolute minimum of heuristic influence. The set of generated solutions should therefore correspond to pseudo-random chromosome mutations that are a direct result of the minimal directed evolutionary pressure applied through this minimal *intelligent direction*.

## 5. SOFTWARE MODEL PERFORMANCE BASE-LINE USING NON-OPTIMISED ALGORITHMS

---

### 5.1 The simulated algorithm types

The semantic and syntactic specification required for a fully functional run-time simulation model derived from the SDC hierarchical class structure has been addressed by the implementation of a pair of intentionally simple mutative algorithms. The dominant design criteria for these simulations is that the search algorithm applied to the 3-dimensional dispersed element problem space is predominantly random in the generation of candidate solutions; within the pseudo-random cyclical limitations that apply to current computer architecture.

By stipulating the algorithms must be predominantly random ensures that their overall operation will be strongly non-optimised, and as a result the directed evolutionary pressure will be minimal during the generation of the candidate solutions. This requires that the active mutation operators, and all algorithmic processing steps must also be designed to specifically minimise the possibility of any intelligent selection to ensure minimal directed evolutionary pressure.

1. *Random Landscape Search*

The shape of the *landscape* fitness peaks and troughs, as defined by the problem space, is partially explored in this simulation. From a known, suitably unfit, starting point the chromosome representing the 3-dimensional Drop Sonde based problem domain is randomly mutated: the result is an iteratively isolated attempt to find the best geometric positional solution that it can. After each iteration the genes in the chromosome are reset to the original initialisation state to ensure that the simulation has no *memory* of the results from previous events, as a result each attempt has an equal probability of optimal fitness.

2. *Random Progression Search*

This simulation type is similar to *Random Landscape Search* in operation, in that the problem space is also traversed by an iterative sequence of random chromosome mutations in an attempt to determine the optimal candidate solution. The active algorithm in this simulation is still essentially random, it differs only in that the chromosome is not reset at any time during the run once it has been initialised: therefore each mutation event forms part of a series of chromosome element mutation events related by the chromosome state of previous iterations. The result is a traversal, or progression, across the problem space that retains the state of the previous event as the start point of the next mutation event. The retention of a history describing all previous mutative steps and also providing the basis for the next mutation step has something of a parallel in one application type of Evolutionary Strategy (ES). The concept of '*cumulation*' where the history of mutative steps actually controls, to some extent, the direction of the subsequent mutative steps as described by (Hansen & Ostermeier, 2001) [54], the idea here is to keep the candidate solutions within the specified confines of the problem space. The *Random Progression Search* does use the previous step

---

definition as the basis of the next mutative step but that is where the similarity ends, the next mutation event is random within the limits of the mutation parameter settings and results in a new set of 3-dimensional coordinates for all elements in the cluster that are derived from their previous values. In addition the problem space confines are also rigidly defined and adhered to but these aspects are controlled, (as with all simulations in this thesis), by the parameters that define the upper-air zone object that contain the elements. These simulation types can be considered as completely random in operation, as directed evolutionary pressure cannot be applied due to the algorithm simply accepting all population chromosome mutations. By sharing the same random mutation operator with the random simulation types makes it appropriate for inclusion in this group of simulation types. The decision mechanism that applies the evolutionary pressure is not particularly strong or directed, but it does have a clearly demonstrable effect on the candidate solution ultimate fitness and the rate of convergence to the ideal fitness condition.

These simulations have all been implemented as a single run-time executable as they share a significant level of structural and procedural code due to the common SDC hierarchy and underlying algorithmic similarity. The shared commonality does allow the option of dynamic run-time switching between algorithms during the search process, this has not been implemented as this could be interpreted as effectively applying some level of heuristic direction to the processing; algorithm switching is applied in a subsequent simulation in a later chapter.

## 5.2 Run-time parameters

The sub-set of the software model parameters, data types and general operation described in this section will be common to many of the simulations used throughout the thesis. The model run-time initialisation file structure and associated parameter descriptions have been described in Appendix C.

As stated above the majority of the initialisation file parameters have settings that are common to many of the simulations in the thesis, including those in this chapter. The initialisation file is in two sections, the first is the Simulation specific settings which is reserved for the externally derived class run-time parameters as defined by the optimisation algorithm and simulation to which they pertain. The second section contains the *generic system* settings of which only a few parameters are of immediate interest, the relevant settings for are listed in figures 5.1 and 5.2 respectively.

Some of the parameters that control the operation of the model have, in effect, become standard and are now the default for use in all simulations. Most of these parameters are defined as part of the VBF/SDC, with the exception of the results statistics and format types which are purely VBF parameters.

## 5. SOFTWARE MODEL PERFORMANCE BASE-LINE USING NON-OPTIMISED ALGORITHMS

---

Parameter	Setting	
<b>Input data-set specification</b>	Single step, maximally randomised inter-element magnitude data	
<b>Fitness function</b>	Minimising differential Sum of squares (inter-element magnitude difference values)	
<b>Zone bound interaction mode</b>	Set element axis to boundary max/min value.	
<b>Results output files</b>	Intermediate compound file Best fitness data Raw fitness data Elapsed Time	
<b>Results file Statistics types</b>	<b>FMT_AVG</b>	Average
	<b>FMT_MAX</b>	Maximum
	<b>FMT_MIN</b>	Minimum
	<b>FMT_SDV</b>	Standard Deviation
	<b>FMT_RGP</b>	Range-groups
<b>Results file format types</b>	<b>FMT_TAB</b>	Tabulated columns
	<b>FMT_TTL</b>	Titles above columns
	<b>FMT_HSTATS</b>	Horizontal calculation
	<b>FMT_FSTATID</b>	Stats Ident to filename
	<b>FMT_STDFHDR</b>	Standard header in file

Figure 5.1: Initialisation file: default parameters

### 5.2.1 Initialisation file section: Simulation specific parameters

In this instance this section reflects the run-time simplicity of the algorithms implemented in this chapter, therefore this part of the initialisation file contains just a single relevant parameter that specifies the type of external simulation type to be applied by the simulation.

The only active operator is a mutation type, as highlighted in bold to signify that its status as the default SDC mutation operator. In summary the **move-element** mutation operator applies a simple random XYZ displacement according to the settings

---

used to configure it prior to the run starting; its derived class structure, functional operation and usage requirements have all been described in detail in the previous chapter. As shown in figure 5.2 all other operators and heuristic methods are not active for these simulations.

<b>Parameter</b>	<b>Setting</b>	
<b>External simulation type</b>	3	Random Landscape search
	4	Random Progression search
<b>Active mutation operator</b>	7	Move-element - default operator (3-D Cartesian position mutation)
<b>Active crossover operator</b>	X	None (not part of algorithm)
<b>Active heuristic mode</b>	X	None (not part of algorithms)

Figure 5.2: Initialisation file: Simulation specific parameters

### 5.2.2 Initialisation file section: Generic system parameters

There are currently over eighty discrete parameters in this section (these are listed in appendix C), of these a significant number can use the default settings and others ignored as the random simulations are both relatively simple in their specification that they have no relevance.

The relatively few generic system settings that are directly relevant are listed in figure 5.3, in addition to a sub-set of the previously mentioned default parameters (highlighted in bold) that are common to all simulations defined in this section.

## 5. SOFTWARE MODEL PERFORMANCE BASE-LINE USING NON-OPTIMISED ALGORITHMS

---

Parameter	Setting
Input data-set standard groups	10, 20, 50, 100, 200 (elements in data-set) **
Population Size	1 (single chromosome)
Runs in simulation	10 (per input data-set element group)
Maximum Iterations	10000/discrete run (if useStopFitVal = false)
Minimum Iterations	1000/discrete run (regardless of stopFitVal)
useStopFitVal	False (0), True (positive/negative value)
stopFitVal	Stops run if fitness $\leq$ stopFitVal
Maximum Iterations	10,000 (discrete run)
Comparison per discrete run	10,000 (Iterations * Population Size)
3-dimensional zone bounds	1000.0 (X), 1000.0 (Y), 1000.0 (Z) (units)
Unit-Cube Normalisation	1 (Zone bounds size correction)
Element-count Normalisation	1 (Input data-set element count correction)
Element Position Initialisation	0.1 - 99% of discrete (XYZ) zone Bounds
Element Mutation Step	0.1 - 99% of discrete (XYZ) zone Bounds

Note. \*\* This range has been chosen to be approximately double the previous one.

Figure 5.3: Initialisation file: Generic system parameters

### 5.2.3 Overview of the plotted results data

For all simulations used throughout the thesis the raw data generated for the default output files has been subject to various forms of post-processing to produce a range of graphical plots, data tables and run-time summaries. There are a number of results types common to all simulations, as well as a number of simulation specific data types. Each of the iterative plot values represents an instantaneous, or *flat* piece of data that precludes any way of identifying how a particular value has been generated in terms of

---

the underlying geometry.

It should be also noted that this particular characteristic will be common to every simulation type presented throughout the thesis. The implication of this is that apparently identical, or substantially similar fitness values are highly likely to represent completely different geometric patterns. For the algorithm in this simulation type this is taken to an extreme as the geometric solution underlying each plotted value is generated from a single event following the reset to the initialisation condition. The focus is on generating relative geometry candidate solutions with the primary objective of finding one that exhibits the minimum, or ideally zero, error fitness; the set of actual Cartesian geometric axis values are only relevant during the fitness calculation part of the process and then subsequently discarded. The calculated instantaneous fitness values can be viewed as a *first derivation* of the Cartesian data describing the geometry encompassed in a candidate solution; in this way a simple fitness value conceals, but directly reflects, the underlying complexity encompassed in the reconstructed 3-dimensional geometry.

## 5.3 Simulation setup

The random search simulations, despite having near identical algorithms, are considered separately as the results from each are quite different. There is a comparative review of all simulation types at the end of the section.

### 5.3.1 Minimising directed fitness convergence

The primary objective of this chapter is to illustrate the underlying software model performance under conditions of minimal algorithmic interference. As the ostensibly random Move-element default mutation operator is the only source of chromosome metamorphosis events for these simulations it is important that this operator is allowed to function in such a way to ensure that the random characteristic predominates. Therefore any aspect of the set-up or input data characteristic that could be construed as applying algorithmic, heuristic or evolutionary pressure is to be avoided.

Each simulation type in this set has been restricted to a single chromosome population to ensure that the results from every discrete run is based on a single source of geometric data. As the simulation types are defined as being devoid of any type of crossover operator there is also little point in having more than one chromosome as there can be no possibility of interaction between chromosomes.

If however, the population were to consist of more than one chromosome then the active algorithm will require some form of selection process for the best fitness value from the population if all chromosomes are considered as part of the same simulation; this is clearly selection based on fitness and not permitted by the restrictive specifica-

## 5. SOFTWARE MODEL PERFORMANCE BASE-LINE USING NON-OPTIMISED ALGORITHMS

---

tion of the simulation definitions.

To conform to the requirement of zero evolutionary pressure the results from each chromosome would have to be evaluated, and output as a separate entity. An inevitable and, in this instance, desirable consequence of the single chromosome restriction is that it ensures zero population diversity, which in turn confirms that the evolutionary pressure from this aspect will be minimised.

The input data-set is a possible source of evolutionary pressure, the random distribution of the Radio-Sonde elements in the input data-set could be such that it becomes the dominant factor in the run-time progress of the simulations that are open to external influence.

This effect can present itself in a positive as well as deleterious way in that the data-set might be particularly easy or difficult to solve, there is no way to tell until it is actually run through the model. To mitigate against this indeterminate influence, it is proposed that all of the 3-dimensional reconstruction simulations in this chapter are run using a set of ten discrete, randomly generated input data-sets comprising elements in the ranges [10, 20, 50, 100 & 200]. The results from each group of ten runs for each of the ten input data-sets are then averaged together and presented as a single set of results, this will effectively remove any bias effects.

### 5.3.2 Overview of the run-time sequence

The run-time operational pseudo-code is shown in figure 5.4. Following the software model initialisation each compound data-set of  $n$  discrete element counts is disseminated into  $n$  sub data-set object representations in the SDC input data-set structure. As this has been directly derived from the VBF input data-file structure this ensures that the Radio Signal Strength Indication (RSSI) packet formatted data appropriate to this problem space is now made available in a format that is *understood* by the generic iterative VBF and derived sub-systems that specify the data access interface; in this instance these are the operator and fitness functionality classes.

The iterative structure built into the VBF then takes over and applies the active operators, fitness calculation and intermediate and end-run data storage and runs the logic in the externally defined algorithm automatically until all iterations are complete or some other externally defined event causes the end of the run or of the simulation itself. Once all runs are completed these are combined into the default set of results files as listed above. The full data-set of results thus comprises a set of files that are specific to an element count size and a particular input-data-set.

---

```

START: Simulation
  LOAD: Initialisation file

  /* Generic system settings: Initialisation Stage */

  SET: Initialise all VBF and SDC Model Parameters
  SET: Population to single chromosome
  SET: Active Fitness Function to ERROR_DIFFERENTIAL
  SET: Active Chromosome Mutation Operator to RANDOM_MUTATE

  /* Simulation specific settings: Algorithm Dependent Selection Stage */

  IF ( External simulation == Random Landscape Search )
    SET: Active Algorithm Object to Random Landscape Search
    SET: Active Virtual Chromosome Selection to NONE
  ELSE-IF ( External simulation == Random Progression Search )
    SET: Active Algorithm Object to Random Progression Search
    SET: Active Virtual Chromosome Selection to SELECT_IF_FITTER
  ELSE ERROR - TERMINATE: simulation
  END-IF

  /* START: Algorithm Run Stage */

  START: Runs (1..n)

    RUN(n): Initialise/Reset Data Collation Sub-system
    START: Run (n): Iteration Phase

      CALL: Virtual Apply Active Algorithm
      STORE: Iterative Results
      END: Run (n): Iteration Phase
      RUN (n): Post-process Results
    END: Runs (1..n)

  /* START: Results Output Stage */
  OUTPUT: Combined Runs (1..n) Results and Statistics files
    WRITE: Population fitness average results files
    WRITE: Best fitness average results files
    WRITE: Raw fitness average results files
  END: Simulation

```

Figure 5.4: Run-time pseudo-code overview

## 5. SOFTWARE MODEL PERFORMANCE BASE-LINE USING NON-OPTIMISED ALGORITHMS

---

### 5.4 Random Landscape Search

For the landscape search simulation the problem space is randomly traversed with the objective of determining the general shape, or topography, of the problem space through the distribution of fitness peaks (or troughs for a minimising function) that are statistically significant when compared with the general level of the plateau of background noise of general *un-fitness*.

```
/*Start Algorithm Run Stage */
START: Runs (1..n)

  SET: Chromosome alleles to random Cartesian positions
  STORE: Initial Chromosome allele Cartesian positions state
  START: Run (n): Iteration Phase
    CALL: Virtual Apply_Active_Algorithm
      APPLY: Active Fitness Function
      APPLY: Active Chromosome Mutation Operator
      APPLY: Active Virtual Chromosome Selection (***)
      APPLY: Reset Chromosome allele to (stored) initial state
    END: Virtual Apply_Active_Algorithm
  STORE: Iterative raw and best fitness results
  END: Run (n): Iteration Phase
  APPLY: Run (n): Post-process Results
END: Runs (1..n)
```

Figure 5.5: Pseudo-code of Random Landscape Search algorithm

In order to maximise the ability of the simulation to *look around* the problem space it is important to negate any effects that the very presence of a software modelling system may impose on the search process. To achieve this there must be a minimum of restrictions applied to the parameters controlling the algorithm. In the Random Landscape Search simulation type the only parameter that can change is the mutation operator, the mode of operation of this operator is common to all elements in the search space through the modification of the Cartesian coordinates representing the XYZ axes.

The first run-time step in the search process, which is identical for all simulation types used throughout the thesis, is the fitness assessment of the search space as it was randomly initialised. Thereafter the algorithm specific to the simulation is dominant. For this simulation the underlying run-time mode is to apply a one-shot approach to a series of random chromosome mutation events where each allele, representing a Radio Sonde element, is moved to a new position within the 3-dimensional zone that defines the *physical* aspect of the search space. Each mutation event is then followed by the

---

objective fitness assessment for the chromosome as a whole, the value being stored as raw data for final output. At this point the indexed allele objects in the chromosome are reset to the values used to initialise the simulation.

This is summarised in the pseudo-code in figure 5.5, which has been partly reproduced from figure 5.4; additional algorithm specific steps are underlined in blue and those not part of the algorithm are marked by (\*\*). The iterative reset event ensures that a new generation of candidate solutions will be generated for each iteration that are related by their shared initialisation condition and nothing else.

As a result of this method each new candidate solution will have a minimal shared geometric relationship with candidate solutions generated by any previous iteration. The reset event removes the need to generate a new set of initial starting coordinates at every iteration, as to do this would, ideally, also require that the new set of coordinates are checked for uniqueness against the run-time historical data to ensure the level of randomness was not overly compromised by excessive repetition.

The application of the pseudo-random Cartesian coordinate position generator that is part of the mutation operator ensures that each set of resultant candidate solutions represents a similarly random attempt to solve the problem; each attempt will have a similar, if quite small, probability of generating an optimal solution.

#### 5.4.1 Random Landscape Search algorithm results

In keeping with the maximally random design ethos of this simulation the main set of plotted results are taken from the *Normalised Raw Fitness Average* set of files. As described in section 5.2.3 this is simply the average of the current fitness values for all chromosomes, as the population for this simulation is set to one then the data in the output directly reflects the *as mutated fitness value*. The plotted averaged and normalised *raw-fitness* results of the Random Landscape Search simulation are shown in figure 5.6. This is the final compound averaged output from ten independent simulation runs that each comprise ten runs of each of the discrete input data-sets; as part of the endeavour to minimise evolutionary pressure.

## 5. SOFTWARE MODEL PERFORMANCE BASE-LINE USING NON-OPTIMISED ALGORITHMS

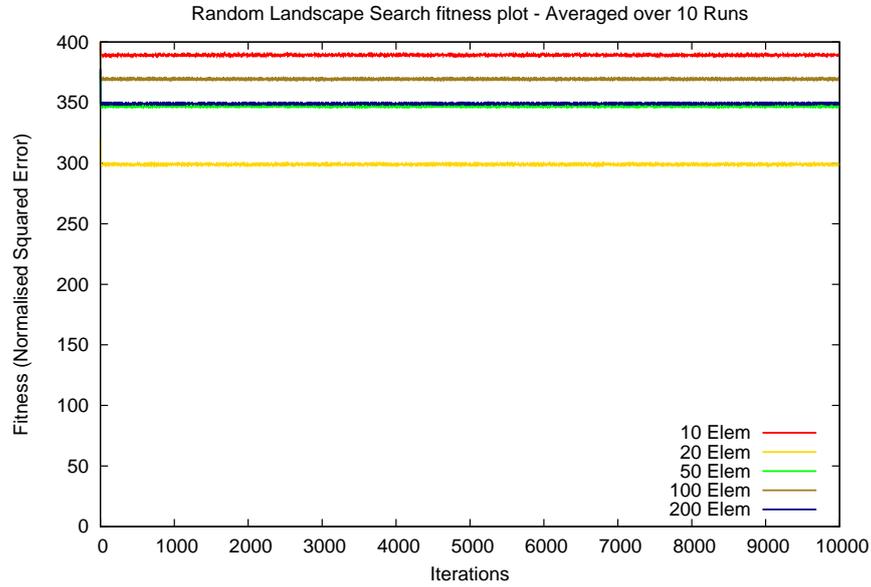


Figure 5.6: Random Landscape Search Fitness Plot

Problem size	PS10	PS20	PS50	PS100	PS200
<b>All-run start fitness condition</b>					
Initialised fitness	422.303	320.588	375.396	400.643	377.835
<b>Terminal fitness</b>					
Absolute Best	235.849	202.510	245.784	208.139	253.175
All-run Mean	389.270	298.944	347.237	369.483	349.187
All-run Worst	501.151	535.529	450.309	458.924	424.118
Average Delta	33.0336	21.6437	28.1587	31.1603	28.648
Average Delta %	7.82225	6.75125	7.50107	7.77756	7.58212
<b>Standard deviation across all discrete runs</b>					
Absolute Best	2.14598	1.24428	0.86098	0.55363	0.43269
All-run Mean	3.42252	1.97145	1.30853	1.01615	0.54788
All-run Worst	4.84625	2.75485	2.14038	1.66354	0.72700
<b>Range Groups</b>					
Group definition	[10,D]; 0≤:<2; 2≤:<5; 5≤:<10; 10≤:<20; 20≤:<100; 100≤:;				
Weighting factors	10.0 1.0 0.1 0.01 0.001 0.0001				
Hits metric	0 0 0 0 0 10	0 0 0 0 0 10	0 0 0 0 0 10	0 0 0 0 0 10	0 0 0 0 0 10

Table 5.1: Random Landscape Search results table

---

On first observation of figure 5.6, the plot traces are virtually flat, the inference being that very little of real significance occurred as a result of processing the Random Landscape algorithm. In terms of the achieving the ideal terminal fitness value of zero this is clearly the case, apart from the initial step change in fitness that occurs during the first few iterations there is an absolute minimum of fitness convergence. The initial fitness step change is significant in that it may indicate that the fitness level, labelled as the *starting condition* in table 5.1, actually represents a substantial level of *un-fitness*, so unfit perhaps that even a totally random algorithm can engineer a measurable change in overall fitness.

If there was a very small level of progressive convergence to an ideal fitness value of zero, then there will also be the possibility that it will have been *smoothed-out* of existence by the process of combining the results from the runs and input data-sets. If this is the case then such a small amount of convergence can be realistically considered as insignificant as it may be an anomalous idiosyncrasy caused by the random nature of the input data-sets. As avoidance of unwanted input data-set influence was precisely one of the main reasons for using multiple input data-sets these empirical observations suggest that this decision has been vindicated to some extent.

The summary of the results for all problem space sizes, (PS10 to PS200), is shown above in table 5.1, in the form of Starting condition to the *Absolute Best* terminal fitness value. The latter is the best, or most fit result that the simulation achieved at any time during the entire set of runs, across all input data-sets for that particular problem space size. This particular set of fitness values is of little use and not actually present on the plotted result of figure 5.6 as it represents a spot-value and not the more statistically significant and reliable overall performance of the simulation as represented by the *All-run Mean*. This value also forms the *Delta* values that show just how much change there has been between the *Start condition* and the *All-run Mean*.

The *Standard deviation* values can be used to provide a quite useful guide, in an empirical sense, in providing a *repeatability factor* to assist in determining a *goodness factor* the overall performance has been, and how much credence can be applied to the *Absolute Best* fitness value. This is an area that is investigated later in this chapter with an alternative system that provides a quantifiable metric to the overall *goodness* of the performance.

In summary it is quite clear from this set of results that the initial *Start condition* fitness value and the step-change that occur after the first few iterations have strongly dominated the ultimate performance of this simulation for all problem spaces (PS10 to PS200). The relatively modest *Percentage Delta* improvement of approximately just 7% over the *Start condition* values is testament to the almost total lack of convergence to the ideal fitness condition.

## 5. SOFTWARE MODEL PERFORMANCE BASE-LINE USING NON-OPTIMISED ALGORITHMS

---

### 5.5 Random Progression Search

The Random Progression Search simulation follows a natural evolution from the Random Landscape Search since they share many structural, algorithmic and run-time characteristics. The problem space is still randomly traversed by an iterative sequence of random chromosome mutations, it differs only in that the single chromosome in the population is not reset at any time during the run once it has been initialised; this is the defining, and only, difference between these algorithms.

This simulation builds a new candidate solution by applying the mutation event to the Cartesian axis positional data stored in the chromosome from the previous iteration; the result is a progression, or traversal, across the problem space related only by the chromosome state of all previous iterations. In common with the previous random algorithm the mutation operator is the only source change to the chromosome, the mode of operation of this operator is also identical. After each mutation event, subsequent objective fitness assessment, and storage of the data, the Cartesian XYZ positional data for all allele indexed objects in the chromosome are kept unchanged to provide the Random Progression, (continuity between mutation events) that characterises this algorithm.

This is summarised in the pseudo-code in figure 5.4, which has been essentially identical to the *Random Landscape Search* pseudo-code in figure 5.5, the line that reflects the algorithmic difference is marked by (\*\*\*\*); additional algorithm specific steps are underlined in blue and those not part of the algorithm are marked by (\*\*\*) .

```
/*Start Algorithm Run Stage */
START: Runs (1..n)
  SET: Chromosome alleles to random Cartesian positions
  STORE: Initial Chromosome allele Cartesian positions state
  START: Run (n): Iteration Phase
    CALL: Virtual Apply_Active_Algorithm
      APPLY: Active Fitness Function
      APPLY: Active Chromosome Mutation Operator
      APPLY: Active Virtual Chromosome Selection (***)
      APPLY: Keep Chromosome allele unchanged for next iteration (****)
    END: Virtual Apply_Active_Algorithm
  STORE: Iterative raw and best fitness results
  END: Run (n): Iteration Phase
  APPLY: Run (n): Post-process Results
END: Runs (1..n)
```

Figure 5.7: Pseudo-code of Random Progression Search algorithm

---

### 5.5.1 Random Progression Search algorithm results

This plot and data tables in this section use the same type of results data (*Normalised Raw Fitness Average*) as for the Random Landscape algorithm as described in section 5.2.3. The average fitness results plot of all ten runs from the simulation from this simulation are shown in figure 5.8, this shape of the plot bears no resemblance at all to that of the Random landscape Search (figure 5.6). This result was quite a surprise considering the similarity of this algorithm and that of the Random landscape of the previous section. By the simply algorithmic action of keeping the current fitness result, regardless of how fit it was in comparison to the previous iteration initially produces a good degree of fitness convergence between the first and (approximately) 400 to 600 iterations.

After the *convergence area*, as the plot plainly shows, the data line turns sharply and then very nearly *flat-lines*; in actuality, after this, the fitness results get slightly worse as the gentle upward curve shows; this is indicative that the algorithm does not describe a simple function. The apparent convergence is, in all probability, more a result of the maximal, randomised, *un-fitness* applied to the generation of the input data-sets than any real optimisation. As a consequence virtually any change to the Cartesian axes data describing the 3-dimensional geometry results in an improvement in overall fitness. In addition a progression type algorithm that builds on the previous geometry that was reasonably fit in the first instance is almost guaranteed to generate an improvement in fitness; but only to the point where the ostensibly random approach starts to fail.

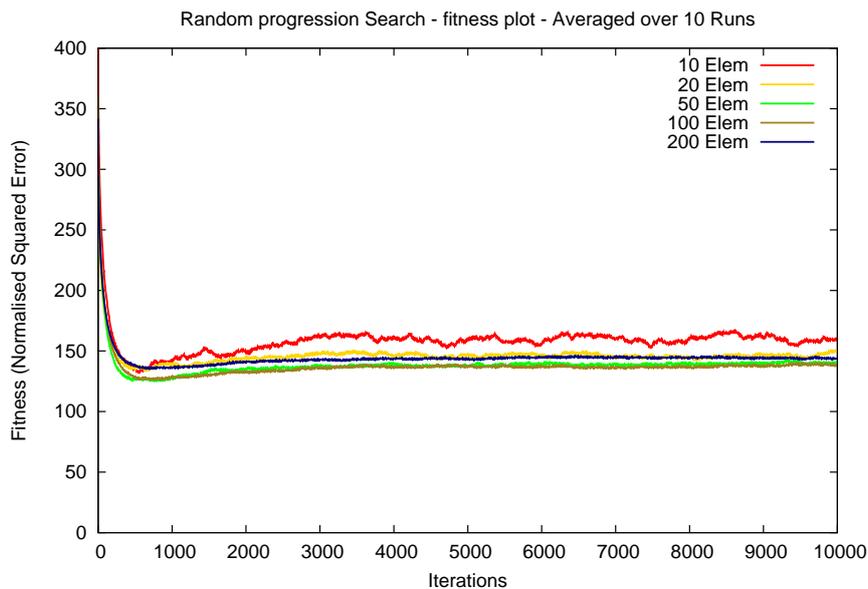


Figure 5.8: Random Progression Search Fitness Plot

## 5. SOFTWARE MODEL PERFORMANCE BASE-LINE USING NON-OPTIMISED ALGORITHMS

---

Problem size	PS10	PS20	PS50	PS100	PS200
<b>All-run start fitness condition</b>					
Initialised fitness	417.284	315.065	348.177	361.762	341.632
<b>Terminal fitness</b>					
Absolute Best	132.676	131.31	112.275	131.645	131.202
All-run Mean	159.422	150.320	140.302	137.616	143.899
All-run Worst	203.234	189.593	163.859	146.287	147.914
Average Delta	257.862	164.745	207.875	224.146	246.608
Average Delta %	61.795	52.289	59.704	61.959	63.959
<b>Standard deviation across all discrete runs</b>					
Absolute Best	4.653	6.304	5.988	3.776	3.878
All-run Mean	29.185	15.310	8.607	6.251	4.940
All-run Worst	47.038	24.047	10.997	10.180	9.595
<b>Range Groups</b>					
Group definition	[10,D]; 0≤:<2; 2≤:<5; 5≤:<10; 10≤:<20; 20≤:<100; 100≤:;				
Weighting factors	10.0 1.0 0.1 0.01 0.001 0.0001				
Hits metric	0 0 0 0 0 10	0 0 0 0 0 10	0 0 0 0 0 10	0 0 0 0 0 10	0 0 0 0 0 10

Table 5.2: Random Progression Search results table

The summary of the results for the Random Progression Search for all problem space sizes, (PS10 to PS200), is shown above in table 5.2. The terminal fitness (***Absolute Best***) shows a distinct change in the overall average fitness values from the original ***Start condition*** values, reflected in the ***Percentage Delta*** values. This is a considerable improvement, although still far from the ideal of ***Percentage Delta*** of 100% signifying the ideal of minimised fitness value of zero.

### 5.5.2 Range Groups and the Weighted Performance Factor

The Range Group and Weighted Performance Factor have been introduced in section 4.6.5, these are simple methods to provide a degree of algorithm performance characterisation that, along with the fitness and standard deviation, can be used in a direct comparison of the results generated by otherwise dissimilar algorithms. The Range Group (RG) data for both simulations depends on range groups that have been set for use with the optimisation simulations that follow in subsequent chapters. All that can be said is that, not unexpectedly, neither of the random simulations performed well enough to generate any useful RG results data, as can be seen in tables 5.1 and 5.2 both simulations failed to produce a single result that would produce anything significant in the Weighted Performance Factor calculations, they are included here for completeness.

---

### 5.5.3 Standard deviation

The standard deviation of the iterative fitness results across all ten discrete runs in a *standard* simulation process is quite indicative of the *stability*, or repeatability of the primary algorithm encapsulated within the active simulation. This data has proved to be usefully indicative when comparing results from pairs of simulations following a change to the active algorithm of one of them.

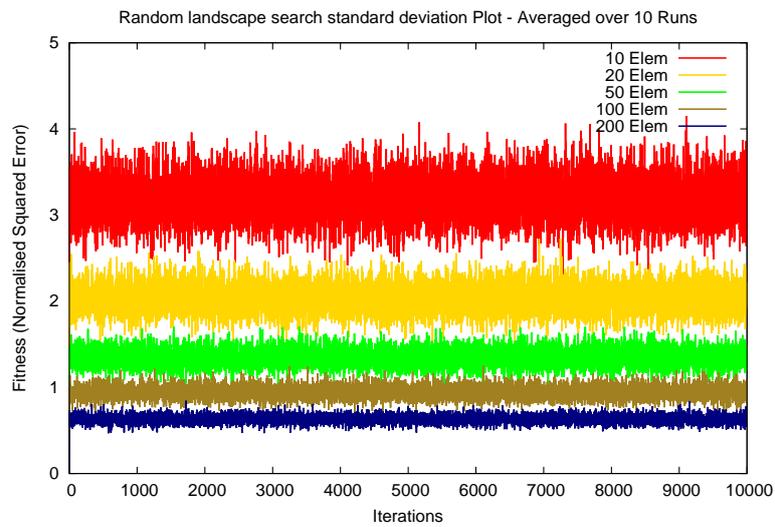


Figure 5.9: Random Landscape Search standard deviation plot

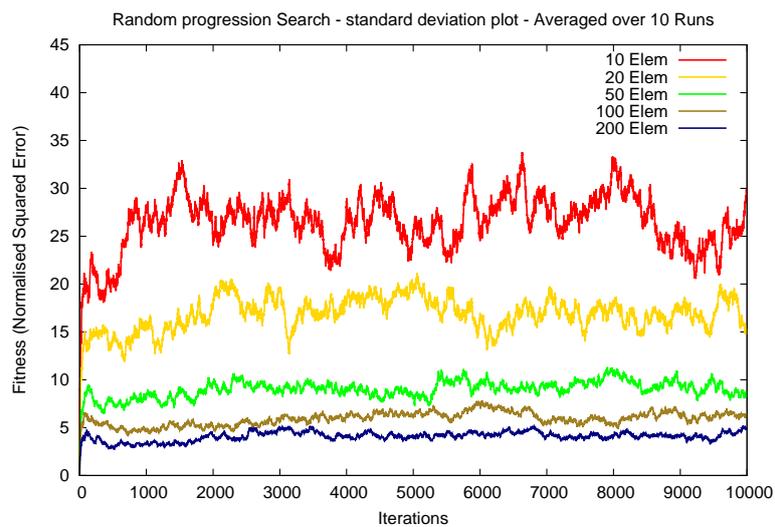


Figure 5.10: Random Progression Search standard deviation plot

## 5. SOFTWARE MODEL PERFORMANCE BASE-LINE USING NON-OPTIMISED ALGORITHMS

---

Essentially, any increase in the magnitude of the standard deviation values indicates a general decline in the repeatability factor; the significance is any algorithm displaying this trait is more likely to produce a discrete run result that is below the average of the whole set of runs in the simulation, thus skewing the overall performance.

The standard deviation results for the Random Landscape Search and Random Progression Search simulations are shown in figures 5.9 and 5.10 respectively. There are several features that are immediately obvious from simple observation for both algorithms.

For the *Landscape Search algorithm*, the plot shows very little deviation around the averaged line value for all problem sizes, this is quite indicative of what may be expected from the *reset to the starting condition* that characterises this algorithm; that the standard deviation is kept within reasonably tight bounds over the entire simulation run indicates that the (VBF/SDC derived) system is exhibiting the highly desirable characteristic of *non-intervention* as there appears to be little evidence of any skewing of the data.

There is just as great a difference between the standard deviation plots of both algorithm types as exhibited by the respective fitness plots. The Random Progression algorithm standard deviation shows far a greater change in the range of values from the start to the point where the values stabilise. This is most evident in the PS10 problem size, it is also interesting to note that occurs at a point that is very similar to the point on the Random Progression algorithm fitness curve in figure 5.8 turned and began to flat-line, being around 400 to 600 iterations.

The population contained just a single chromosome which obviously precludes any chance of crossover of the *genetic* material (a crossover operator is not specified by the algorithm), therefore it can be reasonably expected that the standard deviation would display a large magnitude range; notwithstanding the fact that the standard deviation plot does show the deviation between discrete runs that are never merged, it does suggest that the geometries described by discrete chromosomes are quite dissimilar as would be expected between independent simulation runs.

### 5.6 Summary of the search algorithm results

These two algorithms have almost identical functional properties, yet they have produced results that are quite different, in terms of the overall fitness achieved, the overall shape of the data plots, and the widely differing *Percentage Delta* values of approximately 7.0% and 60.0%.

In the case of the *Random landscape* algorithm the lack of evidence for convergence to the ideal fitness was not unexpected, the iterative reset event ensured there could be no continuity in the data generation throughout the run. The iterative reset

---

event does however ensure that the algorithm generates and assesses the maximum number of, hopefully unique, 3-dimensional geometric candidate solutions potentially contained in the *problem space landscape* using the one-shot approach, as was the intention. As the maximum number of iterations was set to 10,000 and the potential number of solutions being so large as defined in section 4.4.1 there was little likelihood that anything resembling a fit solution would be *chanced upon* in this way; anything relying purely on *chance* is highly unlikely to produce good results, and being random can never be relied upon to be repeatable.

The *Random Progression* simulation was algorithmically, even simpler than that of the *Random landscape* simulation as it lacked the iterative reset event. In this case simplicity worked in the favour of producing better results as the *memory* of previous geometries was kept in the system. This resulted in each iterative generation being based on a previous configuration that had a reasonable chance of being fitter than the original start condition; the outcome of this was to promote a significant convergent tendency in approximately the first 400 iterations, this tendency was clearly absent from the results plot for the *Random landscape*. The *Random Progression* algorithm produced results that were a clear improvement on the *Random landscape* algorithm, but it still ultimately failed to fully converge on anything approaching ideal fitness; again for the reasons suggested above.

In summary, it can be concluded that, based on these results, relying on a purely random approach to generating geometric solutions will be highly unlikely to produce any that are usefully fit. This suggests that the search space problem is sufficiently complex that it can be considered as being NP-Hard, in that it will require some form of optimisation algorithm for it to be solved to any useful degree within a reasonable time-scale. The secondary question was ensuring that the VBF/SDC has no adverse influence on any algorithm implemented with it, these results also strongly support this supposition.

## 5. SOFTWARE MODEL PERFORMANCE BASE-LINE USING NON-OPTIMISED ALGORITHMS

---

## Chapter 6

# Establishing optimal algorithm parameter settings

The objective of this chapter is to establish a standard group of settings for the basic run-time parameters normally associated with Evolutionary Algorithms (EA). The settings have been derived through empirical means with each discrete parameter being the sole subject of a dedicated EA simulation run with all other significant parameters set to reasonable default values.

This process provides a best overall performance setting, and has been the subject of many research papers including (Bäck & Schwefel, 1993) [7], (Michalewicz & Schoenauer, 1996) [85], (Koziel & Michalewicz, 1999) [74], (Ho et al, 2004) [59], (Eiben et al, 1999) [35], (Vesterstrom & Thomsen, 2004) [126], (Varela et al, 2003) [98], (Pereira-Neto et al, 2005) [95], and (Beyer & Schwefel, 2002) [10], each of these projects researched the possibility of characterising a set of parameters specific to their particular research subject. The parameter value set determined in this chapter is also quite specific to this research subject, but there is a strong parallel to the general set of combinatorial optimisation problems as the generic EA has been shown to be applicable to many different classes of problems.

### 6.1 The default parameter values

There are many EA specific parameters that may be considered for this process, in this particular EA implementation there are in excess of one hundred parameters covering all aspects of the run-time process. Fortunately it is a relatively simple task to select a sub-set that directly reflect the philosophy of natural evolution and limit the set of parameters to those with a direct correlation, as these are also most likely to have the strongest effect on the run-time processing; the final set of parameter settings are the average result for all element sub-sets, (PS10 to PS200), within the input data-set, appropriate to this specific problem space EA implementation.

## 6. ESTABLISHING OPTIMAL ALGORITHM PARAMETER SETTINGS

---

The actual value is often selected from a range that offer a similar performance level, or chosen to minimise the amount of run-time processing required. The selected value is therefore likely to be a compromise for one or more of the problem space sizes, the obvious implication is that some or all of the settings will favour, to some extent, some element group sizes more than others; in this instance the emphasis on the selected parameter setting will probably be set to favour the larger element group sizes as these represent the more complex simulations, where this does not overly conflict with the run-time performance requirement. The parameter values, wherever they are appropriate, will be applied to all EA based simulations throughout the thesis, this approach facilitates maximum comparability between related simulations. The parameter list follows.

- Chromosome (number of alleles) mutation distance
- Chromosome population size
- Selection mode type
- Mutation operator type
- Crossover operator type
- Chromosome mutate/accept (unchanged into new population) ratio

Each individual parameter is applied using a set of default settings for all other parameters, these are listed below. These default settings have been derived somewhat arbitrarily, through general observation of how different settings of the various parameter types affected model performance during the software development stages. A set of default parameter values are required for the *'initial'* settings, these are used as the basis for all the following simulations, with any modifications to the settings determined by successive parameter simulation runs applied as required. These initial settings have been selected as they have been observed to have produced reliable and effective run-time results during the software development process; this makes them particularly useful as part of the run-time characterisation process.

The initial parameters are listed below.

<b>Parameter</b>	<b>Default setting</b>
<b>Chromosome mutation distance</b>	3.0% of maximum XYZ bounds
<b>Population size</b>	40 chromosomes
<b>Selection mode type</b>	Tournament
<b>Optional selection parameter</b>	25.0% of population size
<b>Mutation operator type</b>	Move element standard
<b>Crossover operator type</b>	Uniform crossover

---

<b>Chromosome mutate/accept ratio</b>	70.0%/30.0%
<b>Simulation type</b>	Generational EA

Each simulation is a combination of ten runs for each Radio Sonde element group problem size (PS10 to PS200), and for each discrete step in the range of values pertinent to the specific parameter under evaluation. The final value to represent the specific parameter is selected according to the best fitness averaged across all problem sizes as it is quite feasible that one parameter setting may be the absolute best for all.

## 6.2 Mutation distance

This is one of the most significant EA parameters, it is of considerable importance that it is set to a value that is suitable for the problem specification under consideration; an inappropriate setting is likely to hinder the performance quite noticeably. The ability for an EA to converge on a suitably fit candidate solution is directly related to how this parameter is set, both as an initial value and through being modified by any active heuristic in direct response to the emergent run-time performance. For example, if the mutation distance is large then the disruption to the population will also be commensurately large.

In general this will result in poor convergence as suitably fit chromosomes (or part thereof) may not be permitted to stay in the population for long enough to significantly influence the evolution that should result in general convergence. A large mutation distance can however prove useful, when applied sparingly; this mode is frequently referred to as '*anti-stagnation*' as it usually creates a significant disruption to the overall diversity of the population, the benefit to this is to provide a large '*shake-up*' that should result in a greater variance in the '*genetic*' diversity of the population. This will encourage divergence and may also result in an overall increase in the objective fitness of the general population. If the mutation distance is set too low then convergence is likely to also be a slow process, to the extent that convergence may be prevented altogether.

A small mutation distance helps preserve population divergence as the subtle changes to chromosomes that occur as a result of the low rate will tend to preserve fit chromosomes for longer allowing them to play a part in many more evolutionary events. The simulation aims to determine an effective value for the mutation distance that encourages good divergence through maintaining population diversity whilst avoiding excessive disruption through a rate that is set high.

The set of run-time mutation distances is shown in table 6.1.

## 6. ESTABLISHING OPTIMAL ALGORITHM PARAMETER SETTINGS

Mutation distance values (as % of maximum XYZ distance)										
group one	0.5	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0
group two	10.0	20.0	30.0	40.0	50.0	60.0	70.0	80.0	90.0	100.0

Table 6.1: Mutation distances

### 6.2.1 Mutation distance - simulation results

The plotted results from this simulation are shown below in figure 6.1 and in table 6.2. The plot lines show a clear tendency to flatten out between 2.0% and 10.0% mutation distance for the PS10, PS20 & PS50 element group problem space and between 2.0% and 10.0% mutation distance for PS100 & PS200 element group problem spaces. Either side of what may be called the *'best performance plateau'* the average fitness for all problems spaces shows an equally clear tendency to progressively deteriorating fitness values according to the complexity of the problem space.

The plot zones either side of this *'best performance plateau'* (this is between approximately 3.0% and 10.0%) clearly demonstrate a diminished performance, but they have different causes. In the case of the 0.5% to 2.0% mutation distance zone this results in a convergence rate that is probably just too low. For the 10.0 to 100.0% mutation distance zone convergence is unlikely to occur as the large mutation distance causes such significant disruption to the diversity of the population.

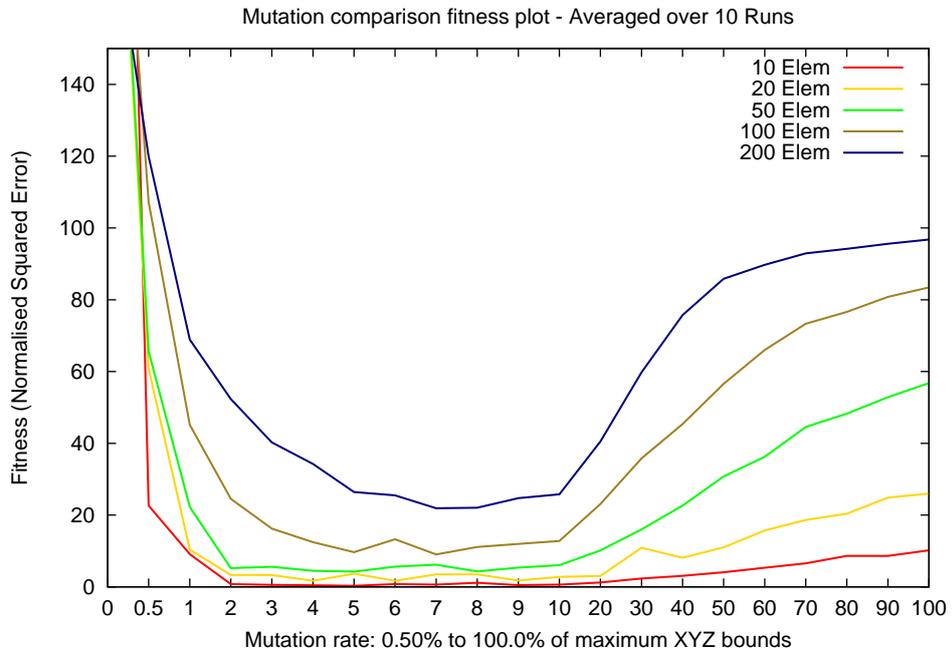


Figure 6.1: Mutation Comparison Plot

---

The effect of the mutation distance on the fitness convergence characteristics indicate that the software model implementation generally fulfils what may be expected of an EA based optimisation algorithm. The plotted data is summarised in table 6.2, the *Selected value* for each element sub-set has been set at the point where it coincides with the *Best fitness value*. The value selected is shown in the *Best mutation* row, this value has been selected to favour the larger problem sizes as indicated previously.

Problem size	PS10	PS20	PS50	PS100	PS200
<b>Best fitness value</b>	0.313	1.705	4.252	9.038	21.911
<b>Best mutation (%)</b>	5.0	6.0	5.0	7.0	7.0
<b>Selected value (%)</b>	<b>7.0</b>				

Table 6.2: Mutation fitness comparison

### 6.3 Chromosome population size

This number of chromosomes in the population will always have a significant effect on the overall performance of any EA based optimisation algorithm. There is a direct relationship between the population size and the potential pool of ‘genetic’ material. This should result in a directly proportional relationship to the potential candidate solution diversity; this assumes that the initial set of chromosomes remains sufficiently diverse throughout the run-time processing.

In the idealised situation the size of the population will always be as large as possible, the drawback of this is the inevitable penalty in the time taken to complete a run-time simulation being proportional to the size of the population. Therefore the objective for this simulation is to determine a minimum setting that still allows good fitness convergence across all the problem sizes.

The range of population sizes in shown in table 6.3.

Population sizes										
Range	10	20	30	40	50	60	70	80	90	100

Table 6.3: Population size

#### 6.3.1 Chromosome population size - simulation results

The results from this set of simulations are shown in figure 6.2 and in table 6.4. The immediate conclusion from the plotted results is that the trend is that the overall average fitness indicates the expected improvement with the increase in the population size. What was quite unexpected is that the fitness values across all problem sizes levels out noticeably after the 50/60 chromosome point. The expectation that the best fitness

## 6. ESTABLISHING OPTIMAL ALGORITHM PARAMETER SETTINGS

would always be at the largest population size was not entirely borne out by some of the experimental results, specifically for the PS10, PS20 & PS50 problem sizes.

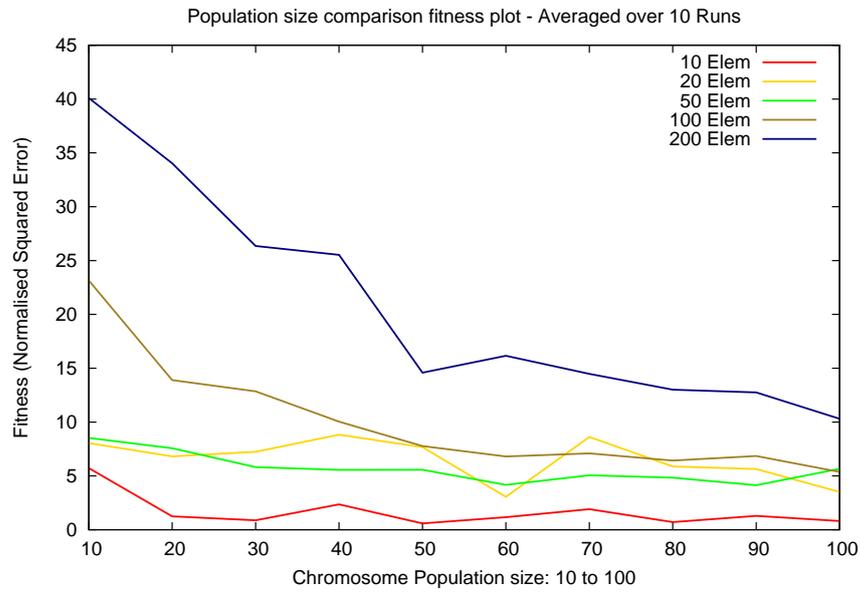


Figure 6.2: Population Size Performance Comparison Plot

Problem size	PS10	PS20	PS50	PS100	PS200
<b>Population Size</b>					
10	5.721	8.0558	8.524	23.149	40.099
20	1.244	6.813	7.574	13.903	34.039
30	0.875	7.242	5.816	12.852	26.352
40	2.353	8.832	5.557	10.046	25.529
50	<b>0.584</b>	7.677	5.575	7.772	14.586
60	1.169	<b>3.058</b>	4.166	6.809	16.158
70	1.910	8.614	5.059	7.089	14.482
80	0.714	5.882	4.835	6.425	13.014
90	1.287	5.633	<b>4.142</b>	6.849	12.757
100	0.813	3.514	5.654	<b>5.365</b>	<b>10.306</b>
<b>Best fitness value</b>					
	0.584	3.058	4.142	5.365	10.306
<b>Best population size</b>					
	50	60	90	100	100
<b>Selected value</b>					
	<b>50</b>				

Table 6.4: Population size performance fitness comparison

---

The largest problem sizes, PS100 & PS200 did produce the fittest results at the maximum population size, and, in general also displayed a progressively improving trend to fitness as the population size increased. Whilst this may appear to favour the larger problem sizes, an alternative view might be that the smaller problem sizes actually do less well because of their implicitly simpler encoded geometries offer less scope for alternatives, with the resultant reduction in diversity across the population; it must be stated that this is conjecture, but does seem quite plausible.

## 6.4 Selection simulations

The software reconstruction model has the facility for a number of chromosome selection modes that can be applied during the run-time modelling phase, these are listed below in table 6.4. The objective of this section is to evaluate the general effect of these chromosome selection methods on the performance of the model software.

### 6.4.1 Tournament selection parameter setting

Of all the available run-time selection modes only the ‘*Tournament selection*’ mode has a parameter that can be set independently, (in the initialisation file), as a percentage of the number of chromosomes in the currently active population. This number is then used to create a ‘*pool*’ of prospective chromosomes from which the replacement is selected using the tournament method at every iteration. The variable parameter can be set between 1.0% and 25.0% (the default setting) of the population size; the 1.0% setting is subject to a minimum of three chromosomes.

To enable the most effective and direct comparison to the other modes it is first necessary to establish the most appropriate setting. This process consists of eight discrete simulations of ten runs using the percentage settings and equivalent chromosome counts.

The range of tournament selection values are shown in table 6.5.

<b>Tournament selection percentage (population: 50)</b>							
Selection %	5.0	7.5	10.0	12.5	15.0	20.0	25.0
Equivalent	3	4	5	7	8	10	13

Table 6.5: Tournament selection percentage and equivalent chromosome count ranges

### 6.4.2 Tournament percentage selection - simulation results

The results from this set of simulations are shown in figure 6.3 and table 6.6. The results shown in the plot and data table clearly demonstrate that the number of chromosomes used in the selection process does not have a very strong effect on the overall convergence on the ideal fitness. The PS200 problem size plot line in figure 6.3 does

## 6. ESTABLISHING OPTIMAL ALGORITHM PARAMETER SETTINGS

---

show a small and relatively constant improvement in the overall level of objective fitness as the number of chromosomes in the selection ‘pool’ increases; all other problems sizes appear largely unaffected by the size of the ‘pool’, it is reasonable to summarise these as showing effects that may be described as being essentially random.

Although this may be not quite as expected, one interpretation of these results is that they do reflect the underlying complexity of the problem search space as described in section 4.4.1. The chromosome, at its simplest interpretation, is a representation of a sequence of real number triples, which when considered in discrete pairs of triples each represent a simple ‘vector-less’ distance error magnitude value between the two points in 3-dimensional space. The implication is that each chromosome is subject to this triple-pairing inter-dependence and that swapping one set of triple-pairs for another may have little effect as the 3-dimensional positional inter-dependence is the dominant factor in determining the overall fitness of the population. Therefore having a larger pool from which to choose the fittest chromosome has the potential to produce only small variations in fitness as the model run progresses.

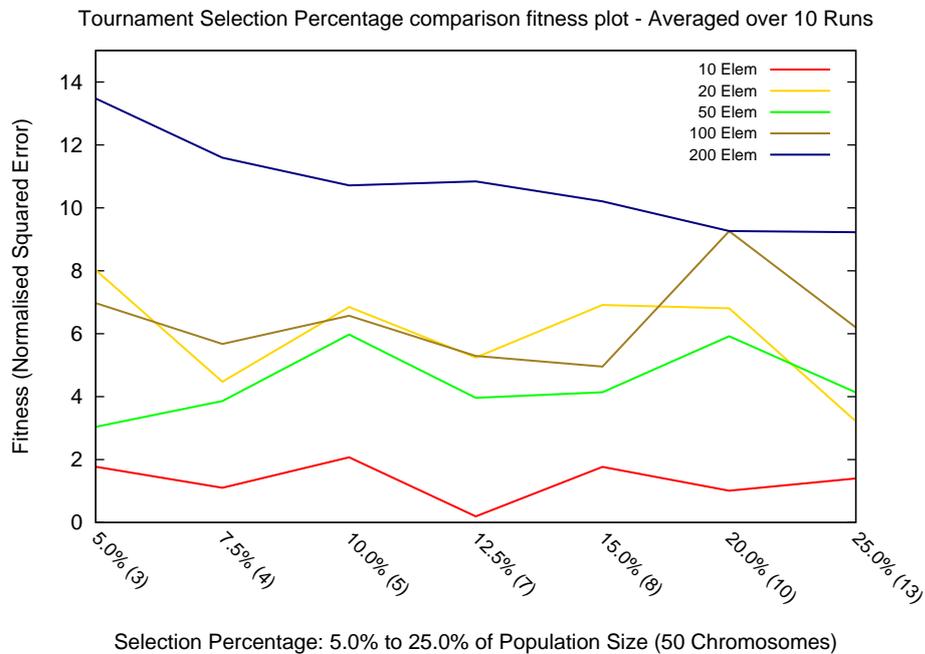


Figure 6.3: Tournament percentage selection plot

---

<b>Problem size</b>	PS10	PS20	PS50	PS100	PS200
<b>Percentage (Chromosome count)</b>					
5.0% (3)	1.77	8.02	3.04	6.97	13.47
7.50% (4)	1.10	4.47	3.86	5.67	11.63
10.0% (5)	2.07	6.85	5.97	6.57	10.71
12.5% (7)	0.19	5.24	3.96	5.29	10.84
15.0% (8)	1.77	6.91	4.14	4.95	10.21
20.0% (10)	1.01	6.80	5.92	9.26	9.26
25.0% (13)	1.40	3.22	4.13	6.20	9.22
<b>Best fitness value</b>					
	0.19	3.22	3.04	4.95	9.22
<b>Best Percentage</b>					
	12.50	25.0	5.0	15.0	25.0
<b>Selected value</b>					
	<b>25.0% (13)</b>				

Table 6.6: Tournament selection percentage fitness comparison

It is also clear from table 6.6 that the lower percentage settings do impose a limit on the best fitness value, as apart from the slightly anomalous best fitness value at 7.50%, all others are at the higher percentages. From this it has been concluded that the most effective setting for the ‘*Tournament percentage selection*’ is 25.0% of the active population size. Although this setting will impose a slight performance penalty due to the additional time taken to select a larger number of chromosomes it is felt that the potential of better terminal objective fitness on the PS200 problem size outweighs this whilst not adversely affecting the remaining problem sizes.

## 6.5 Selection mode type

The objective of this section is to establish the comparative effectiveness of the set of the run-time modes used to select a single chromosome from the current population for subsequent placement in the next generation population. The process of evaluation applied in this section is to perform a set of simulations based on ten discrete runs that exclusively implement one of the listed modes. All other run-time parameters are default settings, or as set by the previous simulations in this chapter including the population percentage parameter for the ‘*Tournament selection*’ mode which is set as described in sub-section 6.4.1.

The selection modes that can be applied during the run-time modelling phase are listed below in figure 6.4.

## 6. ESTABLISHING OPTIMAL ALGORITHM PARAMETER SETTINGS

---

### Selection group type

<b>Tournament selection</b>	Select fittest chromosome from 'n' sized group
<b>Random selection</b>	Select a chromosome at random regardless of fitness
<b>Least Fit selection</b>	Always select least fit chromosome for replacement
<b>Random mode select</b>	Randomly activate selection mode every iteration

Figure 6.4: Selection mode types

### 6.5.1 Selection mode type - simulation results

The results from this set of simulations are shown in figure 6.5 and table 6.7, the 'Tournament percentage selection' mode being set to 25.0% as concluded in sub-section 6.4.1. The format of the plot differ from all other simulations in this chapter in that the diamond data points are discrete elements representing the averaged best fitness obtained by averaging the results from across all ten runs. The plots also use inter-connecting lines as these help illustrate the subtle difference between the data points and are not intended to suggest that the results between selection modes are linked in any other way.

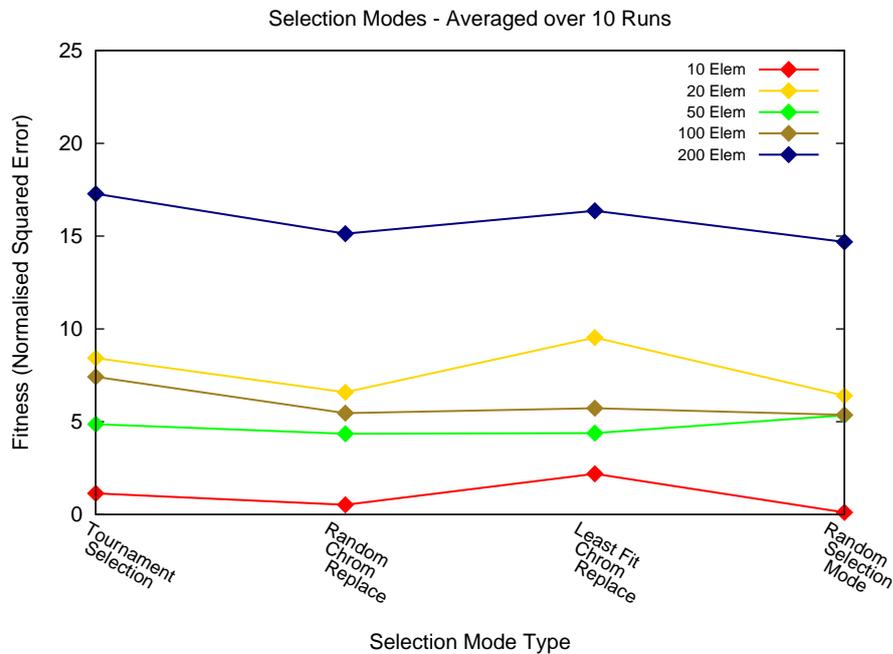


Figure 6.5: Selection mode type comparison plot

---

<b>Problem size</b>	PS10	PS20	PS50	PS100	PS200
<b>Selection mode type</b> - averaged over ten runs					
Tournament selection	1.14	8.43	4.87	7.42	17.30
Random selection	0.52	6.60	4.35	5.46	15.13
Least Fit selection	2.20	9.53	4.39	5.72	16.37
Random mode select	0.11	6.40	5.34	5.26	14.68
<b>Best fitness value</b>					
	0.11	6.40	4.35	5.26	14.68
<b>Selected mode</b>	<b>Tournament selection</b>				

Table 6.7: Selection mode type fitness comparison

An alternative to this method would have been the use of a bar chart, however in order to show the data for all selection modes and for all problem sizes (PS10 to PS200) it would have required a 3-dimensional bar-chart. The visual format of this type of chart proved inadequate when trying to depict the subtle differences between fitness values for all of the modes simultaneously.

The terminal fitness values for each discrete problem spaces are all quite similar for all of the selection modes, however no one selection mode is dominant as the best value (in blue) are spread across all modes. The results from this simulation can be reasonably summarised that the active selection mode does not have an overwhelmingly strong or decisive impact on the terminal fitness across all of the problem sizes; the double random ‘*Random mode select*’ demonstrates the best performance and, for this group of simulations, would be the preferred mode.

However the ‘*Tournament percentage selection*’ has been selected, partly as it has been subject to a slightly more rigorous investigation as a result of the ‘*percentage selection*’ run-time parameter, but also because the presence of this parameter has the potential for a further level of control in subsequent, more complex, simulations, that may prove valuable.

## 6.6 Mutation operator type

There are three main operators in use by the software reconstruction model, with a further three available but not actually utilised to any great degree during the simulations in this thesis, these are shown in figure 6.6. The first three (Swap, Shunt & Invert operators) produced little actual benefit to the general fitness, largely as they do not mutate *properly*, that is they do not mutate the XYZ axes just swap between pairs of elements.

This may have benefits for this problem, or other similar types, for example to improve population diversity (if used sparingly); in the interests of model simplicity however

## 6. ESTABLISHING OPTIMAL ALGORITHM PARAMETER SETTINGS

---

their use was kept to a minimum. The Swap & Shunt operators are included here to demonstrate the relative lack of change to the objective fitness that results when used as primary operators.

### Mutation group type - chromosome indexed-pair swap

<b>Swap section</b>	<b>Not</b> active as a primary mutation operator
<b>Shunt section</b>	<b>Not</b> active as a primary mutation operator
<b>Invert section</b>	<b>Not</b> active as a primary mutation operator

### Mutation group type - chromosome allele positional mutation type

<b>Move element Standard</b>	Randomly modify <b>all</b> XYZ axes for <b>all</b> notes
<b>Move element (Random)</b>	Randomly modify <b>all</b> XYZ axes for <b>n</b> notes
<b>Move element (Window)</b>	Randomly modify <b>all</b> XYZ axes for <b>n</b> element block

Figure 6.6: Mutation operator types

### 6.6.1 Mutation operator type - simulation results

The results from this set of simulations are shown in figure 6.7 and table 6.8. The plotted results clearly demonstrate that the operators that *genuinely* mutate the chromosome do produce the best results for this problem space.

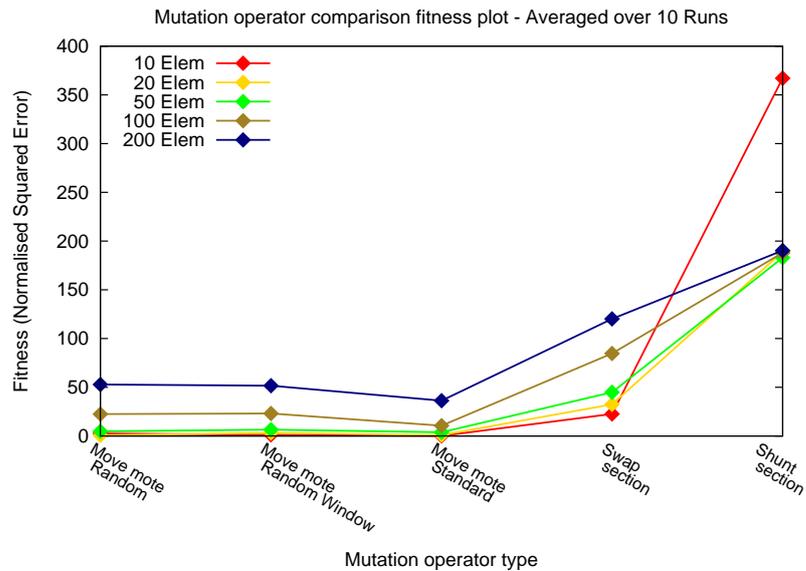


Figure 6.7: Mutation operator comparison plot

---

This was not unexpected, the ‘*Move element standard*’ operator is the main operator used in the majority of the simulations, the other mutation operators (the random & window types) are also used on an occasional, algorithm specific basis.

<b>Problem size</b>	PS10	PS20	PS50	PS100	PS200
<b>operator type</b>					
Move element random	2.603	0.691	4.848	22.483	52.827
Move element window	1.243	3.159	6.55	23.058	51.542
Move element standard	0.114	0.987	3.864	10.548	36.217
Move element swap	22.651	32.473	44.900	84.698	120.139
Move element shunt	367.024	189.096	182.901	187.996	190.296
<b>Best fitness value</b>					
	0.114	0.691	3.864	10.548	36.217
<b>Best population size</b>					
	50	60	90	100	100
<b>Selected value</b>					
	Move element standard				

Table 6.8: Mutation operator fitness comparison

It is clear from the results that the Move element Swap and in particular the Move element Shunt operators do not produce any usable results when used as the primary mutation operator; as this was not the design intention for these operator types, this is neither a surprise, nor a drawback. Their primary purpose was to provide a level of disruption to the population of candidate solutions on an occasional basis, the intention being to produce large changes to a number of elements to increase the diversity of the population. In this they share the function with the ‘*Move element standard*’ operator when it is set to cause a significant *jump* of up to 100.0% of the maximum XYZ bounds to the positions of the elements; this is only used with a very low probability, (less than one percent), and therefore it was not considered necessary to include this as an active operator in this section.

## 6.7 Crossover operator type

The crossover types available are shown below, the operation of these has been described in section 4.5.8. In essence the Single point crossover merges two chromosomes at a single *cut* point, the uniform type merges at a number of these *cut* points. Their effect can be subtle and is frequently problem dependent as to their effectiveness.

In this problem space the generation of new Cartesian positional data for the Drop Sonde elements representations is the primary source of new geometries and therefore any crossover operator is likely to be less effective as their effect resembles that of the Swap, Shunt & Invert operators, albeit the latter only modify a single chromosome.

## 6. ESTABLISHING OPTIMAL ALGORITHM PARAMETER SETTINGS

---

The types tested are listed below.

- Single point crossover
- Uniform crossover

### 6.7.1 Crossover operator type - simulation results

The format for the results in this section is to show the difference in fitness over the entire iterative cycle, this method of plot display is described in detail in section 7.5.2.1. The interpretation of both of these plots is that, as the software model employs a minimising objective fitness function, anything above the zero line is a positive value and therefore represents a less fit result for that iteration, and vice versa; in this instance the Single point crossover is used as the *zero-line* reference data with the uniform crossover providing the difference values that are shown in the plots in 6.8 and 6.9.

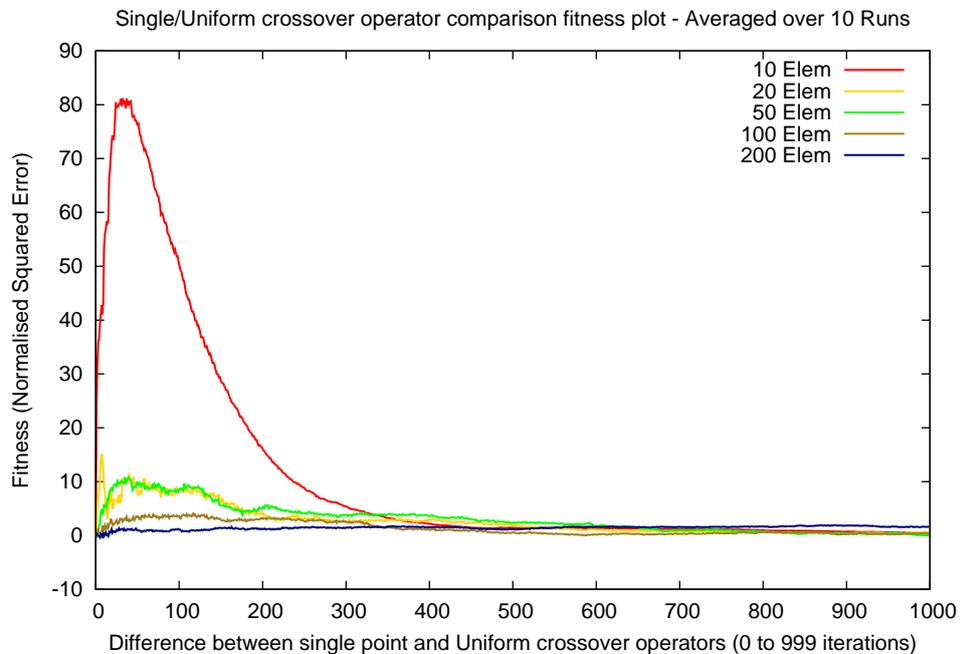


Figure 6.8: Single/uniform crossover difference plot (0 to 999 iterations)

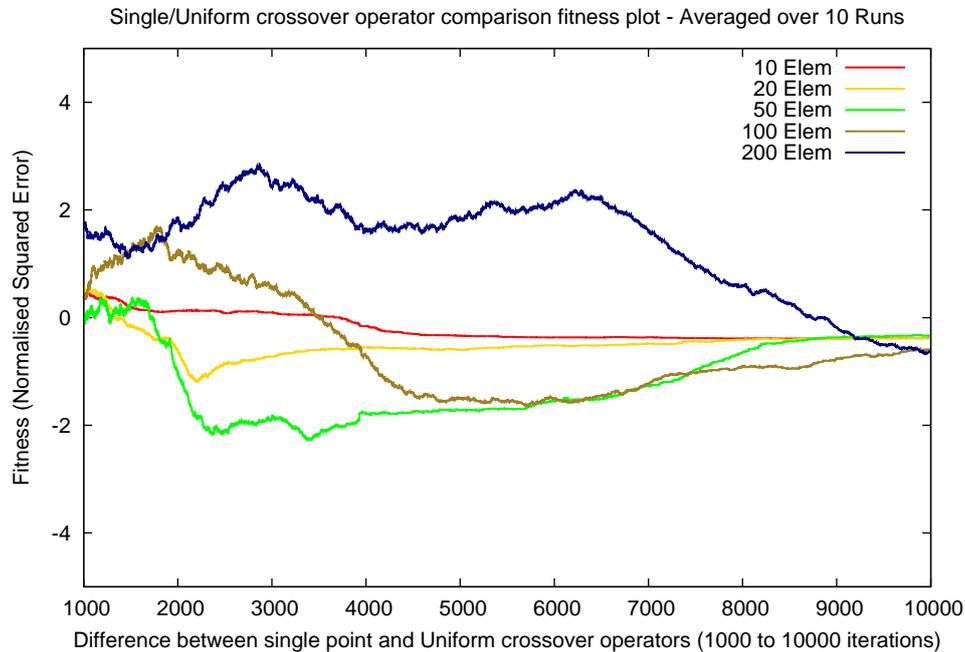


Figure 6.9: Single/uniform crossover difference plot (1000 to 10000 iterations)

The results have been split into two separate plots to allow the illustration of the unusual peak evident in the PS10 problem; to show the entire set of iterations at the Y axis scale required for the PS10 plot-line would have largely swamped the latter iterative differences as these are at a greatly reduced magnitude.

The cause of the PS10 anomaly is, most likely, as a result of the comparative simplicity of the problem space; in the PS10 case there are so few elements that they will be spread throughout the *airspace* zone more sparsely. The latter iterations after 1000 show that the PS10 problem quickly *settles down* and actually displays a much lower difference magnitude than the other problem sizes; this is also likely to be a reflection of the problem simplicity than anything else.

To avoid the PS10 peak, assuming it is repeated for all simulation runs, the Single point crossover would be the better operator, as apart from the PS10 difference, the effect on fitness between the two operators is not greatly significant, and either could be used with little penalty in terminal fitness. The Single point crossover operator has been selected as, in addition to possibly avoiding the anomaly it has a simpler mode of operation that gives a slight run-time performance benefit.

## 6. ESTABLISHING OPTIMAL ALGORITHM PARAMETER SETTINGS

---

### 6.8 Chromosome mutate/accept ratio

This is one of the more fundamental EA operators in that it can have a direct influence on the diversity of the population when using *Generational* type EA. The percentage/ratio setting controls the probability that a parent chromosome will be mutated or kept unchanged in the next generation of chromosome populations across iterations, this ensures a level of continuity and may keep a chromosome that is currently *not very fit* in the current context of the rest of the population; this is the functional opposite of *Elitism* where some of the best chromosomes are always kept.

The benefit to this approach is that genetic material contained in its alleles may become usefully fit as the rest of the population evolves, there is a reasonable probability that what was unfit is now better suited in the current iterative context, it *may* produce a fitter chromosome; there is no way to predict if this will happen however.

The range of values tested are shown in table 6.9.

Chromosome mutate/accept ratio percentage											
group one	10	20	30	40	50	60	70	80	90	100	

Table 6.9: Mutate/accept unchanged ratio

#### 6.8.1 Chromosome mutate/accept ratio - simulation results

The mutate/accept ratio is generally set in favour of mutating the chromosome; the lower the ratio percentage value the more likely the chromosome will be accepted into the next generational population unchanged. As can be seen in the plot in figure 6.10, the problem sizes PS10, PS20, PS50 & PS100 are all largely unaffected by the ratio up until around the 60.0% point where a clear divergence is detected. The PS200 problem space also follows a similar trend with a rather larger fitness range, but the recommended ratio range is limited to between 30.0% and 50.0%. The results for all problem sizes in the 90.0% & 100.0% accept ratio has the effect of reducing any chance of convergence to very low levels, as effectively the mutation is being stopped, or greatly reduced.

The characteristic effect for this parameter appears to reflect the random nature of the EA in general. A range of relatively close fitness results are a feature of the results generated for this parameter, as a result an equivalent range of mutate/accept ratios are shown in the *Best ratio* field in table 6.10. The likely reason for this characteristic is the random nature of EA in general and that the mutate/accept ratio also represents a random effect. The selected value is 30.0% but it could be anywhere between 20.0% and 60.0% with little detrimental effect to the overall performance of the model.

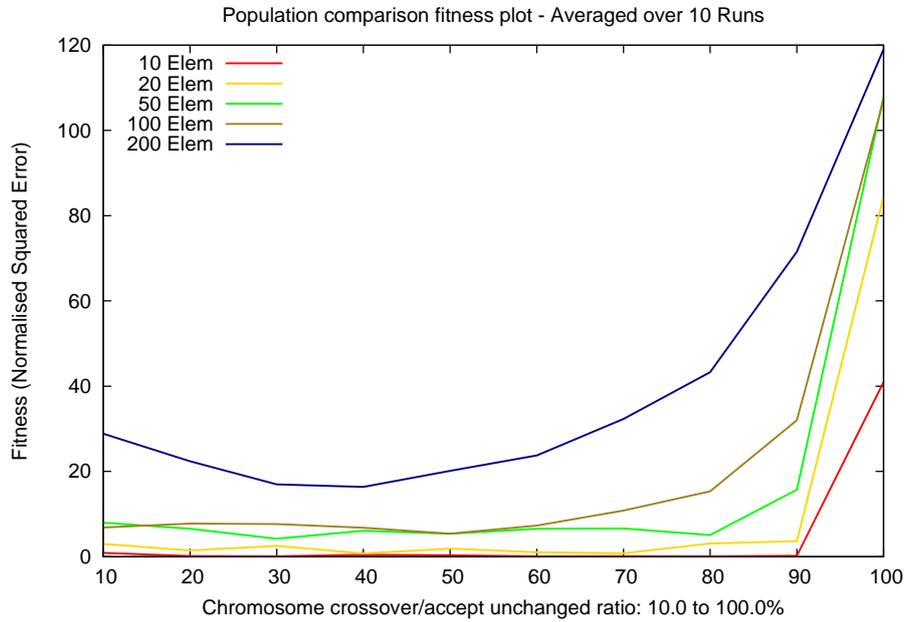


Figure 6.10: Chromosome mutate/accept ratio comparison plot

Problem size	PS10	PS20	PS50	PS100	PS200
<b>Chromosome acceptance ratio (as a percentage)</b>					
10%	0.901	3.009	7.967	6.850	28.852
20%	<b>0.139</b>	<b>1.441</b>	6.549	7.778	22.372
30%	<b>0.088</b>	<b>2.555</b>	<b>4.215</b>	7.663	<b>16.971</b>
40%	<b>0.457</b>	<b>0.741</b>	<b>6.069</b>	<b>6.787</b>	<b>16.375</b>
50%	<b>0.321</b>	<b>1.896</b>	<b>5.374</b>	<b>5.402</b>	20.124
60%	<b>0.096</b>	<b>1.049</b>	6.547	7.302	23.745
70%	0.104	<b>0.788</b>	6.609	10.814	32.332
80%	0.101	3.084	5.062	15.344	43.304
90%	0.219	3.644	15.714	32.037	71.548
100%	40.988	84.543	107.962	107.194	119.038
<b>Best fitness value</b>	0.0888	0.741	4.215	5.402	16.375
<b>Best ratio (% range)</b>	20 to 60	20 to 70	30 to 50	40 to 50	30 to 40
<b>Selected value</b>	<b>30%</b>				

Table 6.10: Chromosome acceptance ratio/percentage comparison

## 6. ESTABLISHING OPTIMAL ALGORITHM PARAMETER SETTINGS

---

### 6.9 Summary

The process of parameter setting has indicated, not entirely unexpectedly, that the values used will be a compromise between the range of problem sizes, with the larger problems being favoured somewhat due to their inherent comparative complexity requiring any assistance available to help minimise the simulation run-time. Additional possibilities are that the settings (individually or as a group) could be ‘scaled’ to the problem size with a simple scalar type approach or be represented as functional pre-sets, for example: ‘*maximum-speed*’ or ‘*best-fitness*’.

The final set of values for the standard group of settings are shown in table 6.11, the *Changed* column indicates if the original default parameter setting selection has been altered as a result of the empirical simulations detailed above.

Parameter	Setting	Modified
Chromosome mutation distance	7.0% of maximum XYZ bound	Yes
Chromosome population size	50 chromosomes	Yes
Selection mode type	Tournament	No
Optional selection control parameter	25.0% of population size	No
Mutation operator type	Move element standard	No
Crossover operator type	Single point	Yes
Chromosome mutate/accept ratio	70.0/30.0%	No

Figure 6.11: Simulation parameters

## Chapter 7

# Optimisation algorithm simulations

The main objective for this chapter is to assess the relative performance of the software modelling system when applied to solving the 3-dimensional geometric reconstruction problem using some form of optimisation algorithm based methods. This is achieved by the implementation of a group of simple optimisation algorithm based simulations using the set of Standard Derived Classes (SDC) directly derived from the Virtual Base-class Framework (VBF) as described in chapter 4.

The first part of the chapter describes two simple optimisation algorithms that have been designed and implemented to employ a strictly limited level of *'intelligent direction'* with the explicit intention of minimising the influence on the evolutionary process. The application of evolutionary selection pressure will generate a set of candidate solutions that, whilst still largely defined as stochastic, will amply demonstrate the positive effect that the *'selection of the fittest available solution'* can have on the rate of convergence.

### 7.1 The simulated algorithm types

The dominant design criteria for these algorithm simulation types was that the search algorithms developed to solve the 3-dimensional dispersed element problem space are still predominantly random in the mutation of the chromosomes. The difference is that the algorithms implemented for this chapter have been modified to include simple selection mechanisms, (appropriate to the algorithm), that can be applied at every iteration to determine the acceptance of any chromosome into the new population. The design of the simulations described here are intended to implement algorithms that have the potential for improved convergent performance, but that are also recognisably an evolution of the simple purely random algorithms of chapter 5.

## 7. OPTIMISATION ALGORITHM SIMULATIONS

---

### 1. *Stochastic Hill-climber (SHC)*

By applying the simplest selection process to the *Random Progression Search* algorithm in chapter 5 a basic SHC algorithm has been implemented. The process encoded in this algorithm is simple, a mutation step followed by the selection process; this is a simple test of chromosome fitness, if the new chromosome is fitter than its predecessor it is always kept, otherwise it is discarded. In practise this is achieved by reversing the mutation event through the reinstatement of the previous chromosome before the next iteration. The SHC algorithm implementation does not include any Simulated Annealing or any way of detecting or compensating for becoming trapped on local troughs or peaks it is likely to be somewhat limited in ultimate performance, however this simplicity may be compensated for in faster operation especially when compared to a more complex algorithm.

### 2. *Evolutionary Algorithm (EA)*

The algorithm implemented here conforms to the minimum requirements that usually define an EA; specifically, build a random chromosome population, assess the population for fitness, followed by mutation, selection, recombination, and subsequent iteration until the defined stop condition is met. The algorithm used here has no defined heuristics or any other means to apply evolutionary pressure other than that normally associated with the implicit functionality of an EA. The intention being to ensure that the operation of this particular EA implementation is as simple as possible. This approach is necessary to prove the fundamental functionality of an EA software model developed using the VBF/SDC, to maximise the comparability with the SHC (and by implication the random algorithms), and finally, to provide an indication of the underlying performance data for direct comparison with subsequent EA simulations that use more advanced techniques.

Following the same process as with the purely random based simulations, the data generated for the default output files have been post-processed to produce similar graphical plots and data tables as described in section 5.2.3. There are additional results types that illustrate and highlight significant differences that successive algorithms produce in the run-time results and an alternative view on the overall performance of the algorithm to generate best-fitness-convergence. The overall objective remains identical, to investigate algorithms and assess their performance on generating relative geometry candidate solutions with a minimum or zero error objective fitness.

## 7.2 Run-time parameters

The majority of the initialisation file parameters have settings that are common to many of the simulations in the thesis, including those in this chapter. Some of the pa-

---

rameters that control the operation of the model have, in effect, become standard and are now the default for use in all simulations. This set is listed in chapter 5, section 5.1.

In summary these settings are: input data-sets consisting of single step, maximally randomised inter-element magnitude data, a fitness function of minimised inter-element magnitude difference sum of squares, results files of multiple run tabulated iterative output of average, maximum, minimum, standard deviation and range-groups (with weighting factor).

### 7.2.1 Initialisation file section: simulation specific parameters

The first part of the initialisation file section defines the simulation specific parameters for the externally defined algorithm, these are completely independent of the VBF/SDC parameters. For the algorithms under consideration there are a total of 27 control parameters, 15 of these are directly relevant to the operation of a ‘*standard EA*’, the remainder being for other algorithms, these are listed in figure 7.1.

The two parameters listed in group 1 define the type of algorithm selected and the active mutation algorithm, for these simulations the standard mutation operator is active (this operator mutates all XYZ axes for all elements); these are the only ones relevant to the SHC, the EA simulation uses these and the remaining parameters listed in all groups. The disparity in the number of parameters between the algorithm types reflects the run-time simplicity of the SHC, and the comparative complexity of the EA simulation.

The set of parameters specific to the standard EA have been further highlighted in blue in figures 7.1 & 7.3. The parameter settings in groups 2, 3 & 4, groups 2 & 3 are felt to be largely self-explanatory in terms of standard EA control parameters, the terminology of those listed in group 4 require some explanation. This group of parameters are designed to increase the diversity in the population by attempting to reduce the potential level of stagnation through the application of some controlled disruption to the population; this is achieved by using the standard mutation operators in a ‘*large disruption mode*’.

These disruptive events are applied by setting the standard mutation operators to apply a large mutation event to a selected number of elements. They are applied infrequently at randomly selected intervals, the size of the mutation event and number of elements thus affected are also determined randomly at each iteration and are described in more detail in figure 7.2. To help ensure maximal population diversity ‘*Elitism*’ is kept switched off, it is not used in any simulation in the thesis.

## 7. OPTIMISATION ALGORITHM SIMULATIONS

---

Parameter	Setting
<i>(Group 1) — Algorithm/operator selection control (Common to SHC &amp; EA)</i>	
<b>External simulation type</b>	3 Evolutionary algorithm
	5 Stochastic Hill-climber
<b>Active mutation operator</b>	7 Move-mote - default operator (3-D Cartesian position mutation)
<b>Active heuristic mode</b>	X None (not part of algorithms)
 <i>(Group 2) — Crossover operator (EA only from this point)</i>	
<b>Active crossover operator</b>	0 Single Point
 <i>(Group 3) — Mutation/selection control parameters</i>	
<b>Chromosome mutate/accept ratio</b>	70% (0.7:0.3 mutate ratio)
<b>Tournament select</b>	25% (Of population)
<b>Chromosome selection method</b>	0 (tournament mode) (0 = tournament replace) (1 = replace random) (2 = always replace least fit) (4 = randomly select methods 0, 1 or 2)
 <i>(Group 4) — Evolutionary pressure control parameters</i>	
<b>Elitism select</b>	0 (False - no elitism)
<b>Allow random element swap</b>	1 (True)
<b>Random swap max</b>	30% (Of elements in active data-set)
<b>Allow random element jump</b>	1 (True)
<b>Random jump max</b>	30% (Of elements in active data-set)
<b>Random jump max distance</b>	99% (Maximum jump distance)

Figure 7.1: Initialisation file: Simulation specific parameters

- 
1. **Allow random element swap.** A boolean setting that enables the swap of the positions of a pair of randomly selected elements. Potentially this can have the effect of moving an element from one extreme side of the modelled air-space to another, the magnitude of the move in 3-dimensional space will depend on the relative Cartesian positions of the selected elements. As stated in the chromosome description 4.5.5 the actual elements (alleles) do not change position in the chromosome as this is an indexed position fixed at the outset as it also defines the unique identifier of the element; the XYZ Cartesian triples are swapped to provide the positional move.
  2. **Random swap max.** This parameter controls the number of element pairs that can be randomly selected (as a maximum) if **Allow random element swap** is set true.
  3. **Allow random element jump** – A boolean setting, this parameter is somewhat similar in operation to the **Allow random element swap** parameter in that it allows a number of elements to make a larger 3-dimensional jump within the bounds of the air-space; the ‘*size*’ of the jump greatly exceeds that of the standard mutation operator and is defined in a following entry.
  4. **Random jump max.** This parameter controls the number of elements that can be randomly selected (as a maximum) if **Allow random element jump** is set true.
  5. **Random jump max distance.** This parameter controls the ‘*size*’ of the jump as a percentage of the maximum XYZ air-space bounds.

Figure 7.2: Initialisation file: Disruption event descriptions

### 7.2.2 Initialisation file section: generic system parameters

The generic system parameters differ slightly from those set for the Random Landscape and Progression Search algorithms in chapter 5. The changes to the generic initialisation file parameter settings are those as determined by the empirical test runs performed in chapter 6, these are shown in figure 7.3, highlighted in blue.

Only the generic system settings directly relevant to the simple SHC and EA simulations have been listed in figure 7.3, these parameters are common to all simulations defined in this chapter.

## 7. OPTIMISATION ALGORITHM SIMULATIONS

---

Parameter	Setting
Input data-set standard groups	10, 20, 50, 100, 200 (elements in data-set) (**)
Population Size	50 (chromosomes)
Runs in simulation	10 (per input data-set element group)
Maximum Iterations	10,000 (per discrete simulation run)
Stop Condition	Run to Maximum Iterations (***)
Comparison per discrete run	10,000 (Iterations x Population Size)
3-dimensional zone bounds	1000.0 (X), 1000.0 (Y), 1000.0 (Z) (units)
Unit-Cube Normalisation	1 (Zone bounds size correction)
Element-count Normalisation	1 (Input data-set mote count correction)
Element Position Initialisation	0.1 - 99.0% of discrete (XYZ) zone Bounds
Element Mutation Step Range	0.1 - 7.0% of discrete (XYZ) zone Bounds

Notes.

(\*\*) This has been set to allow for maximum comparability of results across all runs, regardless of whether an ideal level of fitness has been achieved at an earlier point.

(\*\*\*) This range has been chosen to be approximately double the previous one.

Figure 7.3: Initialisation file: Generic system parameters

### 7.3 Simulation setup

The SHC and EA algorithms encoded in these simulations, although quite similar in many respects, differ sufficiently in the level of directed evolutionary pressure to justify consideration as separate entities. It can be expected that both simulation types will exhibit stronger convergent behaviour than the purely random simulations described previously, and as such, the results from each will also show some level of similarity; a direct comparative review of the simulation types is therefore relevant.

#### 7.3.1 Overview of the run-time sequence

The run-time operational pseudo-code for both simulation types is shown in figure 7.4. The loading of the initialisation file, the data-sets and instantiation of the relevant simulation object types is virtually identical for both of these simulations. Once this has successfully completed the automatic VBF iterative sub-system takes control and applies the externally defined algorithm encapsulated in the active object type. This includes all VBF derived objects; specifically, the airspace zone representation, all active operators, fitness objects and intermediate and end-run data storage.

---

```

START: Simulation
  LOAD: Initialisation file

  /* Generic system settings: Initialisation Stage */
  SET: Initialise all VBF and SDC Model Parameters
  SET: Active Fitness Function to ERROR_DIFFERENTIAL
  SET: Active Chromosome Mutation Operator to RANDOM_MUTATE

  /* Simulation specific settings: Algorithm Dependent Selection Stage */
  IF ( External simulation type == Stochastic Hill-climber )
    SET: Active Algorithm Object to Stochastic Hill-climber
    SET: Active Virtual Chromosome Selection to RETAIN_FITTEST
  ELSE-IF ( External simulation type == Standard Evolutionary Algorithm )
    SET: Active Algorithm Object to Standard Evolutionary Algorithm
    SET: Active Virtual Chromosome Selection to STANDARD_EA_MODE
  ELSE ERROR - TERMINATE: simulation
  END-IF

  /* START: Algorithm Run Stage */
  START: Runs (1..n)
    RUN(n): Initialise/Reset Data Collation Sub-system
    START: Run (n): Iteration Phase
      CALL: Virtual Apply_Active_Algorithm
      STORE: Iterative Results
    END: Run (n): Iteration Phase
    RUN (n): Post-process Results
  END: Runs (1..n)

  /* START: Results Output Stage */
  OUTPUT: Combined Runs (1..n) Results and Statistics files
    WRITE: Population fitness average results files
    WRITE: Best fitness average results files
    WRITE: Raw fitness average results files
  END: Simulation

```

Figure 7.4: Run-time standard EA & stochastic hill-climber pseudo-code overview

## 7.4 Stochastic hill-climber

The stochastic hill-climber (SHC) algorithm described here is intentionally straightforward in design and the level of complexity; as a result it cannot be expected, nor

## 7. OPTIMISATION ALGORITHM SIMULATIONS

---

indeed was it ever intended, to provide the best solution. This implementation can still be defined as being essentially non-deterministic, although it is not entirely random due to the simple decision mechanism used to determine whether to keep the current chromosome or to discard it after every iteration. This method used for the SHC differs in comparison to the algorithmically similar ‘*Random Progression Search*’ (RPS) algorithm detailed in section 5.5 in one significant area. The RPS algorithm ‘*always*’ kept each newly mutated chromosome regardless of the state of the comparative objective fitness between it and that of the previous iteration, the implication being that there can be no selection mechanism applied to this decision for the RP case but is applied in the SHC case. The decision mechanism that applies the evolutionary pressure is not particularly strong or directed, but it does have a clearly demonstrable effect on the candidate solution ultimate fitness and the rate of convergence.

The more common techniques designed to improve the performance of an SHC can be considered as trying to address different aspects of the same SHC algorithmic propensity that become apparent at different points in the process. The first, technique is to develop a method designed to reduce the likelihood of the SHC algorithm becoming ‘*stuck*’ on a local, possibly sub-optimal, maxima (although the term local minima is more appropriate for this problem space and will be used subsequently). This can be achieved by providing a run-time mechanism where an occasional, and large, ‘*disruptive event*’ is applied after a local minima has been detected in an attempt to force the current search to ‘*jump*’ from the current solution onto another area in the search space that is potentially fitter; the mechanism is available for this to be applied but it has not been implemented for the SHC as the intention was to have the simplest algorithm possible.

As a general rule it would appear that SHC algorithms perform best when the iterative mutation step is quite large, however a mutation step that is too large, as with the EA, will cause the SHC to potentially overshoot the best solution local to its current search position, greatly reducing the likelihood of convergence. This is where the second, almost contradictory, technique is useful. This involves progressively reducing the mutation step as the current best solution is being ‘*investigated*’ by the algorithm; more commonly termed *simulated annealing* (SA) due to the progressive cooling process involved in softening some types of metal. The apparent contradiction can be mitigated by careful selection of when to apply large disruptive events, and similarly when to start applying SA, and equally important when to switch back to normal mutative events. However, it is felt that the use of more advanced techniques such as these is more likely to produce better results when applied to the more complex evolutionary algorithm. The main reason for the inclusion of the SHC is partly to further illustrate the flexibility of using the VBF as the basis for similar algorithms, but also to provide additional results that can be used to demonstrate the effectiveness of other algorithms, and hopefully, the improved performance of more advanced algorithms.

---

```

START: Run (n): Iteration Phase

  CALL: Virtual Apply_Active_Algorithm
  APPLY: Active Fitness Function
  APPLY: Active Chromosome Mutation Operator

  /*Evaluate Comparative Population Fitness Stage */
  LOOP: Through Mutated_chromosome population (1..m)
    IF ( Mutated_chromosome[m] fitter THAN Previous_chromosome[m] )
      STORE: Mutated_chromosome[m] IN Previous_chromosome[m]
    ELSE:
      DISCARD Mutated_chromosome[m]
      RESTORE Previous_Iteration_chromosome[m]
    END-IF:
  END-LOOP:
END: Virtual Apply_Active_Algorithm

  STORE: Iterative raw and best fitness results
END: Run (n): Iteration Phase

```

Figure 7.5: Pseudo-code of basic stochastic hill-climber algorithm

#### 7.4.1 Stochastic hill-climber algorithm results

The plotted averaged and normalised *best-fitness* results of the Stochastic hill-climber (SHC) simulation are shown in figure 7.6 and summarised in table 7.1. The plot represents the final compound averaged best fitness output from ten independent SHC simulation runs for problem sizes P10 to P200.

On first observation of figure 7.6 there is clear evidence of progressive and continuous convergence towards the ideal of minimum overall fitness. This holds true from the maximally unfit **All-run starting fitness condition** for all problem sizes (PS10 to PS200). The rate of convergence does diminish but does not completely ‘flat-line’, the plot-lines for all problem sizes track remarkably closely (after approximately iteration 2500), maintaining a fairly stable offset for all problem sizes. Neither does the iterative fitness deteriorate at any point during the process, this is to be expected as the SHC algorithm is ‘programmed’ to only keep solutions that demonstrate an improvement in fitness.

## 7. OPTIMISATION ALGORITHM SIMULATIONS

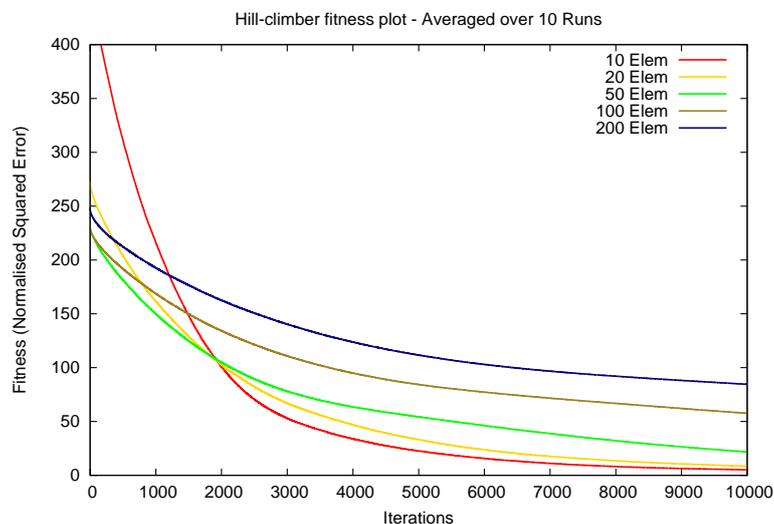


Figure 7.6: Stochastic hill-climber fitness plot

Problem size	PS10	PS20	PS50	PS100	PS200
<b>All-run starting fitness condition</b>					
Initialised fitness	416.066	291.330	270.169	256.614	193.964
<b>Terminal fitness</b>					
Absolute Best	0.376	1.5805	5.226	12.059	28.159
All-run Mean	1.399	2.902	7.865	15.171	35.156
All-run Worst	3.572	7.325	10.939	18.729	43.791
Average Delta	414.667	288.428	262.304	241.443	158.808
Average Delta %	99.663	99.004	97.089	94.088	81.874
<b>Standard deviation across all discrete runs</b>					
Absolute Best	0.179602	1.113983	2.839022	3.361104	3.384986
All-run Mean	1.83313	2.686290	4.465446	5.054473	5.627624
All-run Worst	4.77276	5.205946	5.99972	8.563988	7.67785
<b>Range Groups</b>					
Group definition	[10,D]; 0≤:<2; 2≤:<5; 5≤:<10; 10≤:<20; 20≤:<100; 100≤:;				
Weighting factors	10.0 1.0 0.1 0.01 0.001 0.0001				
Hits metric	8 1 1 0 0 0	6 2 1 1 0 0	0 3 4 2 1 0	0 0 2 6 2 0	0 0 0 1 9 0

Table 7.1: Stochastic hill-climber results table

The PS10 problem size shows a clear initial fitness that is considerably more unfit than

---

all other problem sizes, although this is quickly resolved with the ‘*correct*’ ordering of the results commensurate with problem size; this ‘*apparent anomaly*’ has been noted as a frequent occurrence in many simulation runs during the software development, the cause is felt to be largely due to PS10 having so few elements that they inevitably become more scattered which leads to a increased likelihood that any individual element will be further from its ideal position, with the result that the fitness level is reduced accordingly.

The results summary table 7.1 for the SHC algorithm confirms, numerically, that the terminal performance, (the ‘*Absolute Best*’ field), of the simpler problems sizes (PS10 and PS20) are close to resolving the 3-dimensional geometric reconstruction problem; although there is a clear disparity with the problem size PS50 to PS200 where the terminal fitness is less good. Unfortunately observation of the ‘*All-run Mean*’, the ‘*Hits metric*’ and the standard deviation data clearly indicates that the SHC does not reliably generate good solutions for the majority of the runs. Overall the SHC has produced some good results, quite unexpectedly so given the apparent complexity of the search space in terms of the number of possible solutions.

## 7.5 Standard evolutionary algorithm

The Evolutionary Algorithm (EA) implemented for this section conforms to, what might be best described, as a standard EA configuration, as described by (Goldberg, 1989) [49] in the seminal book ‘*Genetic algorithms in search, optimisation and machine learning*’. common to many implementations such as those by (Duckett, 2003) [32], (Fujiwara & Sawai, 1999) [41], (Kalyanmoy, 2001) [70], (Goldberg, 1989) [49], and (Zitzler et al, 2000) [144] amongst many others. Although the EA as implemented here is quite simple in terms of algorithmic complexity, it implements all of the primary functional attributes, (mutation, recombination, and selection), necessary to effectively evolve solutions appropriate to the problem search space.

In keeping with the underlying non-deterministic characteristic implicit in any EA this implementation cannot guarantee to find the optimal solution, and the results do bear this out as it does get close to ideal fitness in some instances, but does quite not achieve it. The decision mechanism that applies the evolutionary pressure has the potential to provide a stronger level of direction, as in the case of the SHC, this must be applied with care to avoid population stagnation that can occur as diversity can be adversely affected.

Like the SHC algorithm EA can also become ‘*stuck*’ on local, sub-optimal solutions, to help overcome this potential limitation large disruptive events can be applied using the standard mutation operator in **jump** or **swap** mode, as described in section 4.5.7. These are applied totally randomly although it would be simple to implement a stagnation detection algorithm in an attempt to force the current search to ‘*jump*’ from the

## 7. OPTIMISATION ALGORITHM SIMULATIONS

---

current solution onto another area in the search space that is potentially fitter. The simplified overall run-time structure of this EA implementation is shown in figure 7.7.

### **START: Run (n): Iteration Phase**

**APPLY:** Active Fitness Function

**LOOP:** Through chromosome population (0..n-1)

**SET:** parent\_1 = chromosome population[n] (cannot be parent\_1)

**SET:** parent\_2 = **CALL:** tournament\_Select()

**Stage 1 - Select crossover/random unchanged selection**

**IF** ( RANDOM ( 1..100 ) > Active chromosome mutate/accept ratio )

**IF** ( RANDOM ( 1..50 ) < 50 )

**SELECT:** parent\_1

**ELSE**

**SELECT:** parent\_2

**END-IF**

**STORE:** SELECTED parent chromosome

**CONTINUE:** to next chromosome

**END-IF**

**Stage 2 - Apply Crossover to make the new chromosome**

**SET:** child = **CALL:** apply\_active\_crossover ( parent\_1, parent\_2 )

**Stage 3 - Mutate the new Chromosome**

**IF** ( Allow random element swap )

**IF**( RANDOM ( 1..100 ) < Random swap max )

**CALL:** random\_element\_swap()

**CONTINUE:** to next chromosome (no further mutation occurs)

**END-IF**

**END-IF**

**IF** ( Allow random element jump )

**IF**( RANDOM ( 1..100 ) < Random jump max )

**CALL:** random\_element\_swap()

**CONTINUE:** to next chromosome (no further mutation occurs)

**END-IF**

**IF** ( RANDOM ( 1..100 ) < Random swap max )

**CALL:** random\_element\_jump()

**CONTINUE:** to next chromosome (no further mutation occurs)

**END-IF**

**END-IF**

// Standard mutation - only if (low probability) mutations not performed

**CALL:** Apply\_standard\_mutation\_parallelPopV( parent\_1 )

**END-LOOP:**

Figure 7.7: Pseudo-code of standard EA algorithm

---

### 7.5.1 Standard evolutionary algorithm results

The plotted averaged, normalised ‘*best-fitness*’ results of the Standard EA simulation are shown in the main figure 7.8 and summarised in table 7.2. The plots represent the final compound averaged best fitness output from ten independent EA simulation runs.

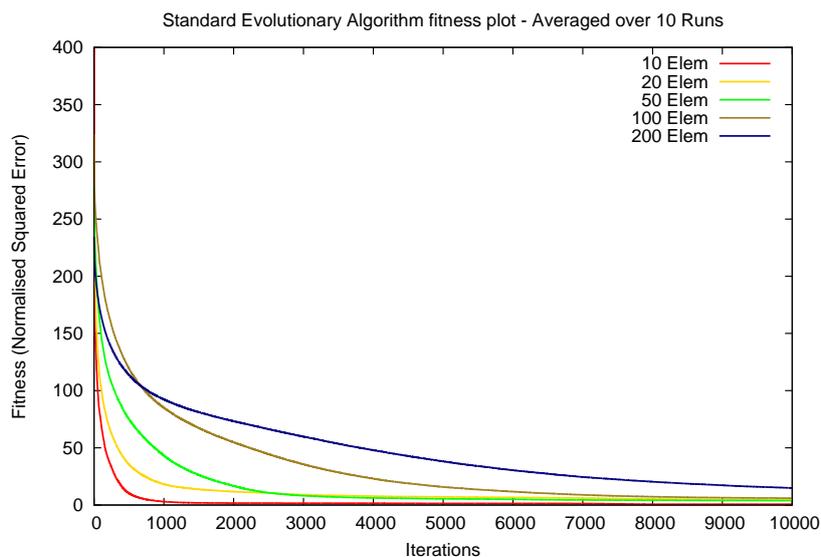


Figure 7.8: Standard EA fitness plot

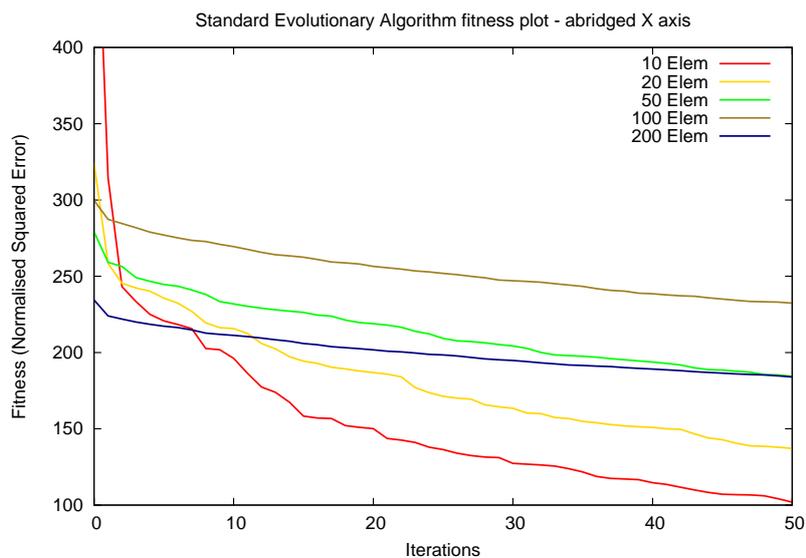


Figure 7.9: Standard EA fitness plot - abridged X axis

## 7. OPTIMISATION ALGORITHM SIMULATIONS

---

As is apparent from figure 7.8 there is clear evidence of progressive and continuous convergence towards the ideal of minimum overall fitness for all problem sizes (PS10 to PS200). The rate of convergence for the first 500 iterations has a steep gradient, the rate for the first ten iterations being particularly so for the PS10 and PS20 problem sizes, this is more clearly seen in figure 7.9 where the  $X$  axis has been curtailed to show the first 50 iterations; this clearly indicates the comparative simplicity of these problem sizes P10 & P20.

Overall the rate of convergence, for all problem sizes, has a defined *'turning point'* at around the 1000 iteration point. That this appears to be true for all problem sizes indicates that the main constraint on convergence is not entirely due to the complexity of the problem; although as a generalisation the rate of change at the *'turning point'* becomes progressively more open indicating that the problem complexity remains a strongly influential factor; again this can be more clearly seen in figure 7.10 where the  $Y$  axis has been shortened to accentuate the visibility of the lines in the plot.

The terminal fitness for the PS10 problem size indicates it has been very nearly solved by the standard EA, although there is clearly room for improvement. When combined with a standard deviation figure of less than 1.2 this is good evidence that the model can reproduce good results across many discrete runs; this suggests good potential for model stability.

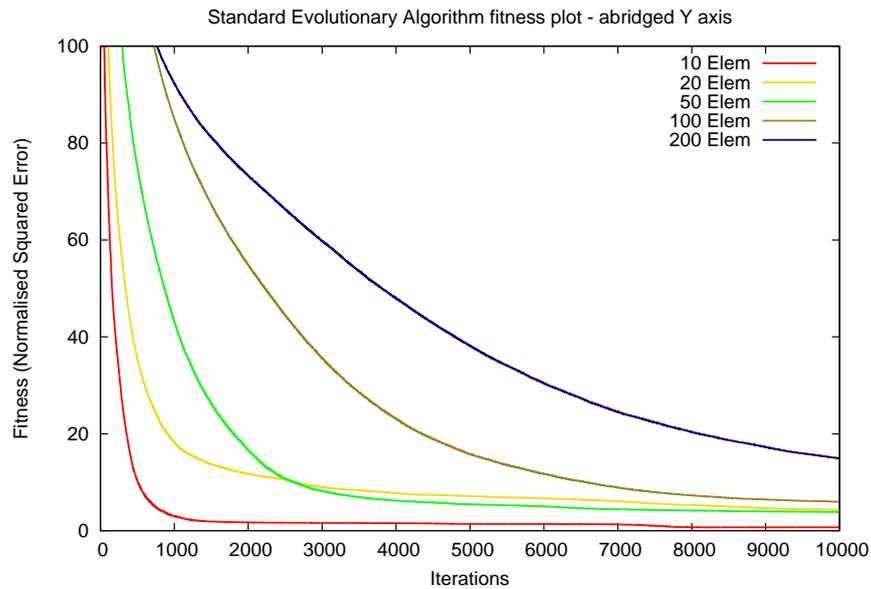


Figure 7.10: Standard EA fitness plot - abridged Y axis

Problem size	PS10	PS20	PS50	PS100	PS200
<b>All-run starting fitness condition</b>					
Initialised fitness	557.260	324.363	279.141	299.746	234.464
<b>Terminal fitness</b>					
Absolute Best	0.457	4.250	3.629	5.234	13.313
All-run Mean	0.684	4.291	3.865	5.962	14.936
All-run Worst	1.587	4.301	3.924	8.875	15.342
Average Delta	556.57	320.07	275.27	293.78	219.53
Average Delta %	99.877	98.677	98.615	98.012	93.629
<b>Standard deviation across all discrete runs</b>					
Absolute Best	1.08571	3.5833	1.95274	1.836163	4.17802
All-run Mean	1.18623	3.70979	2.22905	2.6597	4.84321
All-run Worst	1.58829	4.21579	2.29813	5.9539	5.00949
<b>Range Groups</b>					
Group definition	[10,D]; 0≤: <2; 2≤: <5; 5≤: <10; 10≤: <20; 20≤: <100; 100≤:;				
Weighting factors	10.0 1.0 0.1 0.01 0.001 0.0001				
Hits metric	8 2 0 0 0 0	4 2 4 0 0 0	2 4 4 0 0 0	0 3 6 1 0 0	0 0 2 6 2 0

Table 7.2: Standard EA results table

The PS20 to PS100 problem sizes are less good in terms of terminal fitness and also show progressively worsening standard deviation performance indicating that a lower degree of confidence can be placed in the ability of this version of the model to reliably reproduce comparable results across many runs. The PS20 problem size in particular has produced results that are noticeably less good than that of the PS50 problem. This becomes most apparent from the 1000 iteration point where the rate of convergence is visibly more restrained, especially when compared to the PS10 plot line, from this point the convergence is very nearly a flat line and terminates with a lower objective fitness than the PS50 problem.

This is clearly not the expected outcome and represents an anomalous result given the relative complexity of the PS20 & PS50 problems. The results indicate skewing of the results data-set as a direct result of some of the discrete run data being markedly less fit than others; as the data-sets represented by the plot lines are all averaged from ten discrete runs to have such a strong and consistent effect on the level of convergence the anomalous runs would have to have been exceptional. Analysis of the standard deviation (SD) data in table 7.2 for the PS20 simulation run clearly supports this suggestion as this data indicates a best SD value of 3.5833, this is not in the expected range when compared to the equivalent SD values for the PS10 & PS50 problem sizes

## 7. OPTIMISATION ALGORITHM SIMULATIONS

---

which have values of 1.08571 and 1.95274 respectively; a value closer to 1.80 for the PS20 problem may be reasonably expected.

Further analysis of the terminal objective fitness values and associated standard deviation values for problem sizes PS10, PS20 & PS50, for each discrete run, are shown in table 7.3. This data clearly provides good evidence that the discrete runs objective fitness data for all featured problem sizes exhibit a wide range of terminal values; rather less so for PS10, but it is still present. The PS20 results have a particularly wide range indicating that the data-set used had an encoded 3-dimensional geometry that was difficult to decode if the evolution proceeded away from a particular, and probably, unusual geometry as there four good results of 1.0 or less.

Run	0	1	2	3	4	5	6	7	8	9	—
<b>Size</b>	<b>Discrete run objective fitness (rounded values)</b>										<b>Std dev</b>
<b>PS10</b>	0.08	3.54	0.06	0.07	0.24	0.14	0.18	0.07	0.09	0.09	1.08571
<b>PS20</b>	8.25	0.35	7.97	6.32	1.01	0.26	9.16	4.93	0.34	4.43	3.5833
<b>PS50</b>	6.79	5.82	2.33	1.9	5.36	1.96	4.42	4.15	2.59	0.96	1.95274

Table 7.3: PS10, PS20 & PS50 all-run fitness and standard deviation comparison

Essentially the implication is that for the less good results, (the majority), the evolutionary process resulted in the ‘*wrong*’ local optima became dominant in the population with the resultant low average objective fitness and large SD values. There has been little empirical evidence of that such occurrences were predominant during the entire development stage of the software model, it is always a possibility in a system largely dependent on random events. Since there are no absolute guarantees that any EA will be able to achieve any reliable level of certainty then these apparently anomalous results can be expected to occur on a random basis.

In summary the PS20 input data-set used here seems to be a particularly difficult geometric problem to solve. Although the data-sets for this work have been randomly generated, by effectively using the software model in a reverse mode, it does indicate that, (not surprisingly perhaps), there will always be some problems that are more difficult to solve than others. Obviously the level of difficulty cannot be known until an attempt has been made to solve it, this is compounded in the real-world situation where the data-set is being generated ‘*on-the-fly*’, probably with many RSSI errors, this represents the real challenge for the model to solve. However the overall results are still quite encouraging, and certainly indicate that the problem is not insurmountable.

---

## 7.5.2 Comparison of simulation results

The objective of this section is to compare the results obtained by the SHC and EA simulations and the Random Progression Search (RPS) from section 5.5, chapter 5. The method is to directly compare the performance between two selected algorithms on a point-by-point, or, iterative basis; this comparative method demonstrates, clearly and unambiguously, the (positive or negative) difference between respective algorithms as each run had progressed.

The intention is to clearly illustrate the rate and level of objective fitness for each algorithm and how these parameters evolved over the course of the simulation run. In addition the direct comparison mode between two different algorithms illustrates which was superior at any given instant which may have great significance if a hybrid mode is proposed between the compared algorithms.

### 7.5.2.1 The format of the difference plots

In order to visually demonstrate the relative performance between the complete objective fitness results of two comparable algorithms it is convenient to display them simultaneously on a single plot or chart. This can be achieved quite readily by post-processing two data-sets to create a single difference data-set that directly reflects the numeric difference of the objective fitness values throughout the entire run. The format of this type of plot is therefore very simple and is illustrated in figure 7.11, essentially the plot is now divided into two regions (three if the zero-line representing equality is included). All data-points that appear above the zero-line are greater in magnitude, and those below lesser in magnitude; the designation of base-line and target data-sets and subsequent interpretation of the data are application dependent.

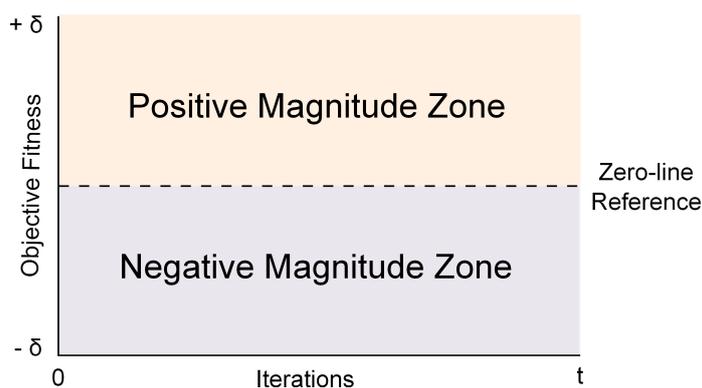


Figure 7.11: Difference plot magnitude zones

Each plot is created from the combination of the run-time data-sets generated by two discrete simulation runs from algorithms applied to the same problem space and, ideally,

## 7. OPTIMISATION ALGORITHM SIMULATIONS

---

each data-set must contain a comparable number of data elements. These data-sets are designated the base-line and target data-sets respectively, the base-line data-set provides the zero-line reference data-points to which the target data-points are directly compared. The zero-line reference values represents zero in the sense that it is the point at which the switch from a positive value to a negative one occurs. Therefore this line does not automatically represent the value of zero at any given point although it is a possibility depending on the data-points in the base-line data-set; in a sense the zero-line data-point values are not of particular importance and may be considered as secondary to those of the resultant difference plot lines.

The plot is read in the following manner: the zero-line represents the discrete objective fitness value achieved by the base-line algorithm at that particular iteration, and for each discrete problem size. This and every subsequent iterative data-point for the target algorithm is subtracted from the equivalent base-line data-point and the result is plotted as a sequence that resembles a continuous curve, (again, for each discrete problem size).

For this particular problem space the objective fitness function is operating in a minimising, or tending-to-zero mode. To translate this into the terms of the difference plot as used here, if a specific problem size plot line data-point is below the zero-line then this represents a better performance for the target algorithm over that of the base-line algorithm as it must be a negative number when compared to the base-line data at that iteration, (ie. following the subtraction step a negative number results which must therefore be tending-to-zero in absolute terms), conversely anything above the line is positive and therefore is less fit. This allows a direct comparison between simulation types, this is particularly useful for determining, in an empirical manner, if a modification to a simulation has resulted in an improved performance.

### 7.5.2.2 Difference plot comparisons: RPS and SHC

This fitness difference comparison plot compares the RPS and SHC with the RPS designated as the base-line data-set, this is shown in figure 7.12. Although the RPS simulation results from section 5.5 could not be said to have produced a particularly strong result in terms of the terminal objective fitness, however the RPS algorithm did produce a noticeable level of initial convergence which makes it useful for the relative performance assessment of the SHC.

As can be seen quite vividly in figure 7.12 the first 3000 iterations the difference data-points for all problem sizes are all in the positive magnitude zone indicating that the RPS has a clear performance advantage over the SHC target data-set; the complexity of the problem size being directly attributable to the point at which the switch to the negative magnitude zone occurs through the gradient. Although the SHC does produce *'better'* results as the difference data-point plot enters the negative magnitude zone on all problem sizes, the outcome of the initial stage of the comparison plot is not what

---

might have been predicted for an apparently ‘stronger’ algorithm.

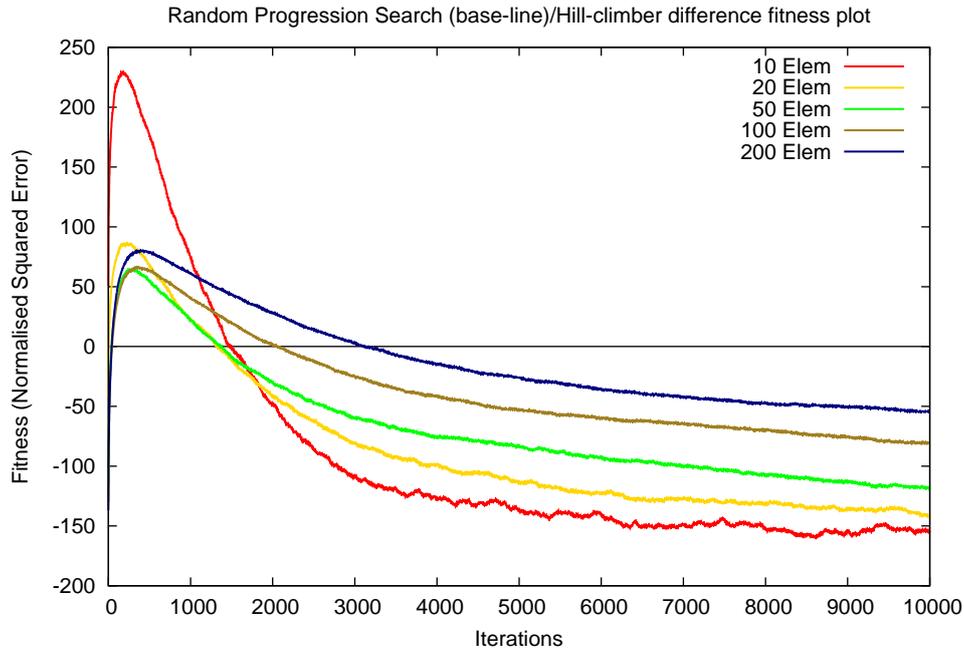


Figure 7.12: Random progression search (base-line)/Hill-climber difference fitness plot

This result is somewhat surprising and some examination of the algorithms is required in order to better understand these results. The first consideration concerns the input data-sets used, in order to maximise the comparability between algorithms the input data-sets used are all identical and have the same initialisation parameters at their creation and subsequent use; therefore the RPS and SHC simulations have both been started from identical initialisation geometries for all elements, for all problem sizes, and for all runs. Therefore the starting condition can be eliminated as a possible cause.

One of the main conclusions from section 5.6 was that the software framework could not itself produce a consistent convergent system, the fact that the RPS did produce some level of initial convergence was considered to be due to the decision to make the initialisation as ‘unfit’ as possible by clustering the elements where they were most unlikely to be located; thus virtually guaranteeing that any mutation will likely produce a more fit chromosome. This state was achieved by applying some control to the creation and subsequent use of the data-sets, when the data-sets were created the elements were allowed to be scattered throughout the ‘airspace’ zone, when they were initialised in the simulation runs, prior to recreating the geometries, they were forced to be a closely grouped cluster thus increasing the chance of maximal ‘un-fitness’.

## 7. OPTIMISATION ALGORITHM SIMULATIONS

---

If the maximal ‘*un-fitness*’ argument is correct then any effects are likely to be as a result of the algorithms themselves and their respective settings. For the RPS all relevant parameters were set at the maximum possible values to help maintain the maximum level of random activity by avoiding any form of evolutionary pressure by restricting the level of the mutation events. The SHC, as previously stated, is quite a simple implementation with a single setting for the maximum mutation any discrete event may apply to the elements in the geometry. When combined with the simple ‘*always keep if better*’ selection mechanism it would appear that the conclusion may simply be that the pressure this applied simply slowed the evolutionary process such that it fell behind the ‘*free-running*’ RPS. That is until the limitation of the random nature of the RPS became naturally limited by the level of chance and probability that had less of an effect on the SHC with its associated selection method. It is quite feasible that a more advanced SHC with a more flexible approach to the level of mutation available at any given point would produce a better result; another solution may be a hybrid algorithm as it is known the RPS is fast by virtue of its simplicity, this could be coupled with a method of deciding when to switch to the SHC for the completion of the run.

### 7.5.2.3 Difference plot comparisons: SHC and EA

This fitness difference comparison plot compares the SHC and standard EA with the SHC designated as the base-line data-set, this is shown in figure 7.13. This result conforms rather more to expectations. Apart from the first few iterations (less than 500 for all problem sizes) where the SHC has a clearly steeper gradient, the EA has a markedly stronger performance. An indication of the relative simplicity of the PS10, 20 & 50 problem sizes is indicated by that fact that the difference magnitude values for these are close to ‘*flat-lining*’ from around 500 iterations and the objective fitness difference magnitude for both algorithms is close to zero at the end of the run.

Whilst the EA still clearly has the performance edge in terms of the terminal objective fitness for the PS100 & 200 problem sizes even the simple SHC implementation does run it surprisingly close. The standard EA used here demonstrates a strong performance on initial convergence across all problems sizes, this level of convergence continues but does tail off considerably, especially on the PS200 problem. It is clear from this difference comparison result that the EA is clearly superior in overall objective fitness performance and confirms that an EA is the more suitable method to solve the geometric reconstruction problem.

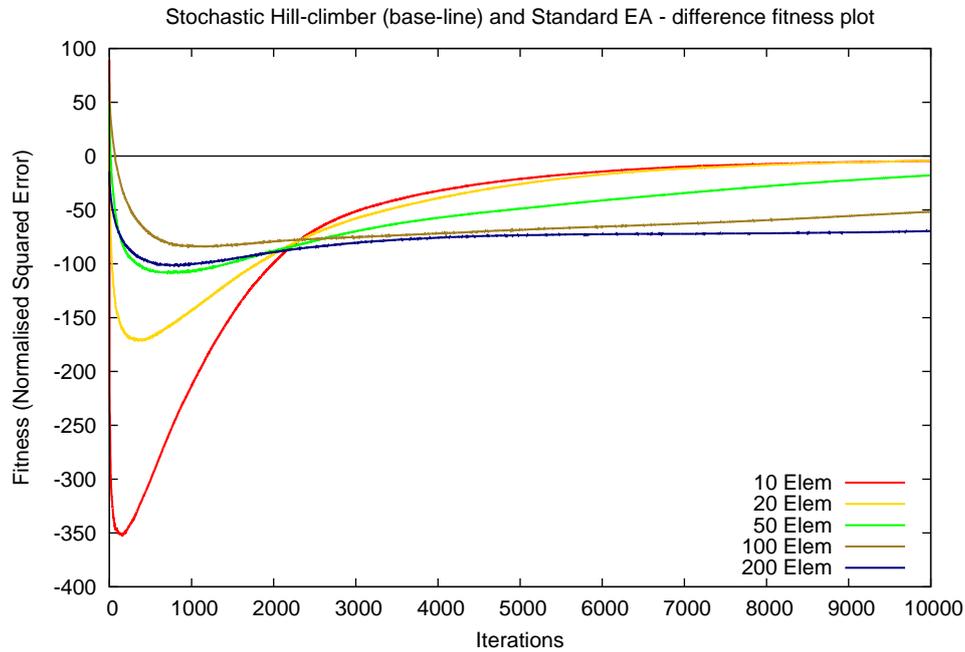


Figure 7.13: Hill-climber (base-line)/standard EA difference fitness Plot

## 7.6 Large data-set evolutionary algorithm results

This final section illustrates the performance of the current standard EA on large problem sizes, between PS250 and PS500, it has been included here for completeness. As can be seen in the plot figure 7.14 and data table 7.4 the overall performance still shows a good level of convergence, but with a commensurately higher, that is less fit, terminal fitness. The time taken to generate results for these large problem sizes precludes them from being included in further work at this juncture.

## 7. OPTIMISATION ALGORITHM SIMULATIONS

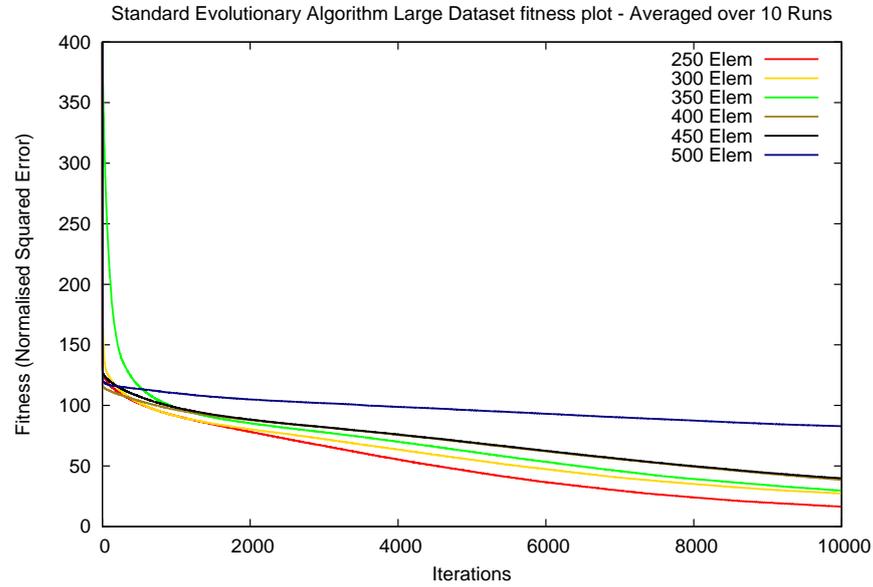


Figure 7.14: Standard EA Large Dataset Fitness Plot

Problem size	PS250	PS300	PS350	PS400	PS450	PS500
<b>All-run starting fitness condition</b>						
Initialised fitness	496.83	471.25	485.61	485.30	479.98	493.03
<b>Terminal fitness</b>						
Absolute Best	10.870	20.151	20.398	33.867	32.999	42.394
All-run Mean	17.16	27.49	28.79	38.99	38.59	80.64
All-run Worst	26.01	39.79	39.19	42.71	45.13	108.42
Average Delta	485.96	451.11	465.21	451.43	446.99	450.64
Average Delta %	97.76	95.53	95.62	92.49	92.62	90.59
<b>Standard deviation across all discrete runs</b>						
Absolute Best	0.6198	1.4177	0.7178	0.2401	0.9074	0.7578
All-run Mean	5.749	4.626	3.7127	2.255	2.719	16.90
All-run Worst	7.978	9.211	14.569	3.202	4.831	30.966
<b>Range Groups</b>						
Group definition	[10,D]; 0≤:<2; 2≤:<5; 5≤:<10; 10≤:<20; 20≤:<100; 100≤:;					
Weighting factors	10.0 1.0 0.1 0.01 0.001 0.0001					
Groups (6)	0 0 0 7 3 0	0 0 0 0 10 0	0 0 0 0 10 0	0 0 0 0 10 0	0 0 0 0 10 0	0 0 0 0 4 6

Table 7.4: Standard EA - Large Dataset

---

## 7.7 Chapter summary

The results of the stochastic hill-climber and, in particular, the standard EA clearly indicate that the problem can be successfully represented, and to an extent, solved. The algorithms presented here have been selected to further demonstrate that the Virtual Base-class Framework and associated derived classes can successfully implement a number of different algorithms using the same underlying software structure.

the primary objective of this chapter was to demonstrate the process of simulating the reconstruction of 3-dimensional geometries describing the position of the modelled free-flying elements. The final conclusion from this chapter is that the base-line data-point results presented here will provide a good basis by which further algorithm developments may be directly compared.

## 7. OPTIMISATION ALGORITHM SIMULATIONS

---

## Part IV

# Search Space Reduction & Heuristic Simulations



## Chapter 8

# Heuristic and hybrid operators

The objective of this chapter is to investigate the possibility of identifying specific characteristics implicit in the run-time framework of the software model and the abstraction of the real-world part of the overall problem space. It may then be possible to translate some of these detail characteristics into a strategy that can be implemented as a heuristic or hybrid optimisation algorithm. The primary outcome of this process will be to investigate the overall improvement to the performance of the software model in terms of the overall run-time speed of operation and, principally, the objective fitness attained throughout all iterations as well as the terminal value.

The process involves identifying certain characteristics implicit in the problem space definition and to consider their potential as sources of additional information that may be useful when solving the geometric reconstruction problem. The first part of the chapter is primarily concerned with the first stage of the development of the heuristic and hybrid concept. The final section briefly introduces some further techniques that were investigated as a result of the work on the initial stage detailed here.

### 8.1 Search space characteristics

The main objective of simplifying any search space is to develop methods that will ultimately generate a statistically significant improvement to the overall objective fitness for the current problem search space. The obvious ideal is that this will be achieved without an excessive increase in the time taken for the active algorithm to reach its best level of convergence. It is also feasible that a reduction in this run-time may occur if the search space simplification method results in a significant reduction in the number of candidate solution permutations to be traversed in the candidate solution search process; with the obvious proviso that the process does not eradicate candidate solutions that may contain suitably fit solutions.

### 8.2 The simulated algorithm types

The dominant design criteria for these simulations is to develop some simple heuristics that can enhance the overall performance of the reconstruction model whilst still retaining the predominantly random mutation of the chromosomes.

1. ***The Gateway Constraint Operator EA simulation*** The algorithm encoded in this simulation investigates the effects of constraining the position of the elements in the cluster to within defined boundaries. The design intention of the *Gateway Constraint Operator* (GCO) is to attempt to limit the search space in order to promote faster convergence against time.
2. ***Hybrid operator simulations*** This simulation type combines the GCO and the *standard* mutation operator used throughout the thesis thus far. The objective is to establish if the order in which they are applied will affect the overall convergence rate and the terminal fitness.

### 8.3 Run-time parameters

The majority of the initialisation file parameters will retain the settings that have been used in all of the simulations up to this point in the thesis, these are as listed in section 5.2.2. In summary these settings are: input data-sets consisting of single step, maximally randomised inter-element magnitude data, a fitness function of minimised inter-element magnitude difference sum of squares, results files of multiple run tabulated iterative output of average, maximum, minimum, standard deviation and range-groups (with weighting factor). Therefore the software model initialisation settings listed in figures 7.1 and 7.3 in chapter 7 are all relevant to the simulations used here, with the exception of the **External simulation type** setting.

### 8.4 The gateway constraint operator

The Gateway Constraint Operator (GCO) makes use of one of the most significant characteristics of the Multi-element Cluster Drop Sonde (CDS). The search space as modelled for this work has a specific requirement that one CDS element acts as overall element cluster controller and data transfer gateway, as described in section 4.1.2. This ‘*gateway*’ element has the required characteristic that it is ‘*fixed*’ at a known reference point in 3-dimensional space; although this itself is relative being subject to the flight path of the monitoring aeroplane, the latter will be able to provide the ultimate absolute 3-dimensional location by virtue of onboard GPS. The combination of a particular operational trait of the real-world Drop Sonde cluster implementation and, certain coincident, discrete data-set elements have the potential of producing a condition that can be used as the basis of a novel type of mutation operator.

---

For this operator the first step is to re-examine the real-world operation of the CDS, during the deployment phase of the CDS the elements are likely to remain in close proximity for at least the first few seconds, although they will spread to some extent as the air currents that they are intended to monitor will cause physical displacement. The amount of physical spread of the cluster is such that it can be expected to occupy a fairly small volume (as a whole), when compared to the physical distance to the gateway Drop Sonde element that, it is expected, will remain on the deploying aircraft. Meanwhile the elements in the CDS at this stage are transmitting their synoptic data messages in the *round-robin* mode described in section 4.1.2.

The suggestion is that the RSSI data between the (*fixed*) gateway element and the dispersed CDS elements share the characteristic that each value, (actually a pair of magnitude values in both transmission directions between the active CDS element and the gateway), ostensibly, represent a similar distance magnitude between each discrete CDS element and that of the gateway element at that moment in time. This particular data can therefore be decoded in a number of ways. Firstly, the gateway to discrete element distance, although not actually known, is directly represented by the RSSI value, and could be used to provide a relative calibration factor specific to each discrete element. The second is that this distance magnitude data also represents the potential for applying a distance restriction on each discrete CDS element at that specific time interval; this is the mode in which it is used here.

#### 8.4.1 Constraint constraint operator: design

The simplest interpretation of the RSSI input data-set is that it is simply a set of numeric values that provide an indication of the magnitude of the distance between two points; in the idiom of the problem space this has been termed the *radial constraint distance*. If one of those points occupies a fixed position (the gateway) then the magnitude values between that fixed point and all other points can now also be considered as being in a fixed relationship that remains valid for that time interval. The problematic underlying nature of the RSSI data-set has been described in chapter 3, as a result the distance magnitude relationship can never be defined completely accurately; there will be a degree of *fuzziness* that will be ever present.

The GCO has the primary function of enforcing a restriction on the set of possible 3-dimensional positions that any discrete CDS element can occupy (potential data-set errors notwithstanding) in such a way that the defined distance magnitude constraints between the gateway element and the remote CDS element can be dynamically maintained by the operator. When the position of an element is mutated, its new position will always be moved towards (and eventually restricted to) the spherical surface defined by this *radial constraint distance*. This is shown graphically in figure 8.1.

## 8. HEURISTIC AND HYBRID OPERATORS

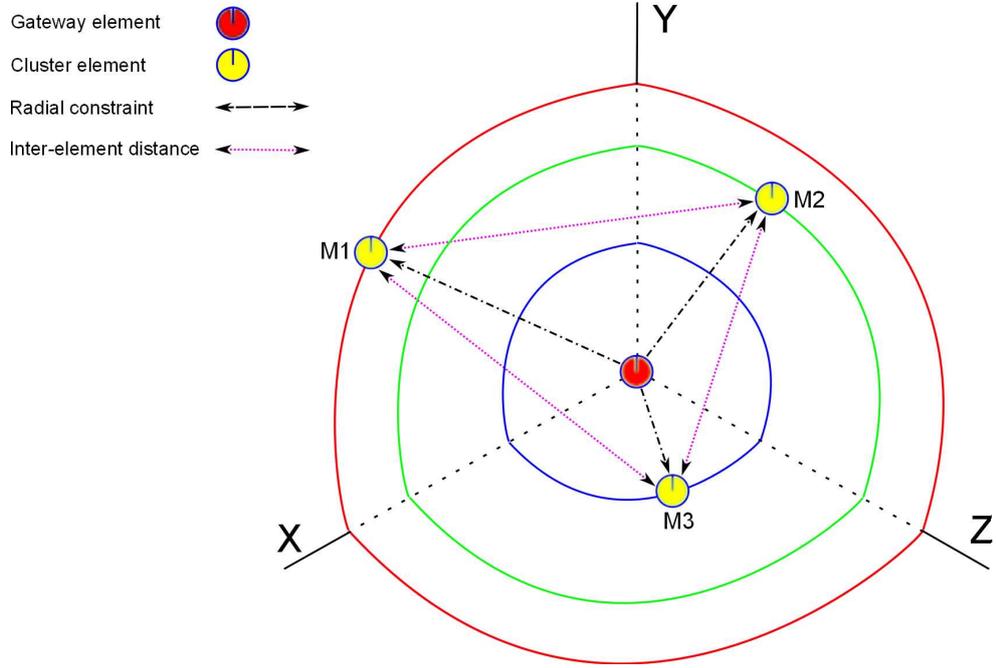


Figure 8.1: Gateway constraint multiple element depiction

Due to the limiting effect of this operator the ostensibly *correct* position of a discrete CDS element should now be somewhere on the spherical surface as defined by the *radial constraint distance* when it is extended to all three Cartesian axes. As a direct result this equates to a significant reduction on the number of 3-dimensional geometries that are now actually possible due to the requirement that the distance magnitude constraint between the gateway element and the CDS element must be maintained. More simply stated, any 3-dimensional geometry that encodes a 3-dimensional CDS element position that is **not** on, or very close to, the *radial constraint distances*, (as depicted as the blue, green & red ‘*spherical surface segments*’ for the three CDS elements M1, M2 & M3 respectively as depicted on figure 8.1), cannot be correct and therefore may be discarded from the search space.

This represents a significant reduction in the size of the search space as many previously possible, and now erroneous, 3-dimensional positions that a CDS element may be mutated to occupy have been eliminated as it is no longer possible for them to lie on the notional spherical surface and to satisfy the *radial constraint distance* heuristic. A simple visual analogy is that of a sphere, the volume and surface area of a sphere are given by  $Vol_{Sphere} = \frac{4}{3}\pi r^3$  &  $Area_{Sphere} = 4\pi r^2$  respectively; this results in an approximate 4 : 1 magnitude ratio in favour of spherical volume for a given radius. In terms of the problem search space used here this equates to a reduction to approximately  $\frac{1}{4}$  of the total 3-dimensional volume to be searched if the search space now consists of only that covered by the spherical surface described above. In actuality the reduction is

slightly more in favour of the GCO restricted search space as the ‘*non-restrained*’ zones (shown as the ‘*out of range*’ stage 1 zones in figure 8.2), extend beyond the spherical surface area which is not included in the volume equation given above. The upper (ie. external) limit now being given by the XYZ cube that defines the 3-dimensional air-space zone used here, the spherical surface cannot extend past these limits.

However, although this technique has great potential for producing a fit result more quickly than just relying on completely random mutation, consideration must also be given to the very real possibility that areas of local *un-fitness* are just as widespread in this constrained space as anywhere else; applying a heuristic operator should improve the probability of finding a good solution but in a truly non-deterministic problem space it is therefore equally unlikely that optimally fit results can be guaranteed.

#### 8.4.2 Constraint constraint operator: Mode of operation

The implementation for this mutation operator, as might be expected, follows the same principle as for all other operators used in the model, the requirement is that they must be derived from the VBF defined operator base class described in section 4.5.6 to be functionally and syntactically compatible. On initial application of this operator the position indicated by the RSSI data between the gateway element and the CDS element could place it at any of the three stages of the operator depending upon the distance magnitude relative to the specific *radial\_constraint* pertinent to that element; these stages, and their relative positions, have been visually summarized in figure 8.2. Subsequent calls will result in the CDS element position being progressively positionally *upgraded*, by being moved closer to the *radial\_constraint* until it finally arrives at stage 3.

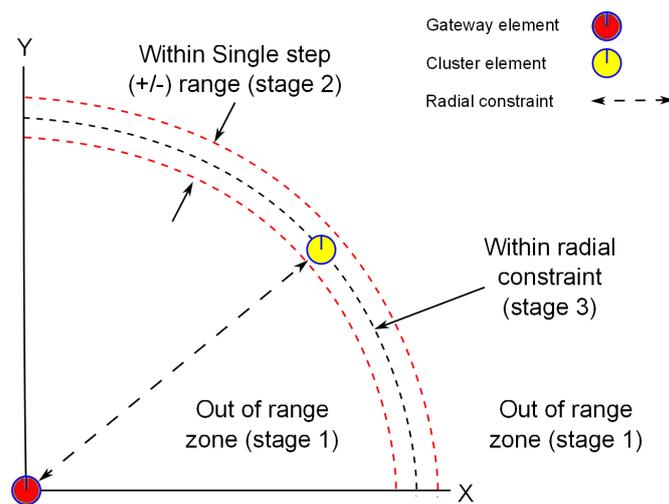


Figure 8.2: Gateway constraint regions

## 8. HEURISTIC AND HYBRID OPERATORS

---

The pseudo-code for illustrating the overall functional structure of this operator is shown in figure 8.3. This mutative effect of this operator is applied to each discrete CDS element in turn, with the obvious exception of the gateway element.

```
IF( radial_constraint VALID )
  Calculate current_distance to GATEWAY for current CDS element
  IF ( within radial_constraint )
    ( Constraint Stage 3 - maintain range )
    Move element → MAINTAIN radial_constraint
  ELSE-IF ( WITHIN max_single_step_mutation )
    ( Constraint Stage 2 - radial_constraint possible in one step )
    Move element → RESTRICT to max_single_step_mutation
  ELSE
    ( Constraint Stage 1 - Progressively move element closer )
    IF ( current_distance < radial_constraint )
      Move element → INCREMENT_XYZ_POSITION
    ELSE
      Move element → DECREMENT_XYZ_POSITION
    END-IF
  END-IF
ELSE (Constraint operator not active)
  Move element → RANDOM (Standard mutation operator)
ENDIF
```

### **Constraint Stage 3 - maintain range**

```
SELECT two RANDOM axes
MUTATE selected axes → RESTRICT to radial_constraint
MUTATE Non-mutated axis to RESOLVE radial_constraint
```

### **Constraint Stage 2 - *radial\_constraint* possible in one step**

```
SELECT two RANDOM axes
MUTATE selected axes → RESTRICT using max_single_step_mutation
MUTATE Non-mutated axis to (possibly) RESOLVE the radial_constraint
```

### **Constraint Stage 1 - *radial\_constraint* NOT possible in one step**

```
INCREMENT/DECREMENT - move Cartesian position linearly
RANDOM displacement value → RESTRICT to max_single_step_mutation
MODIFY X, Y & Z axes by displacement value (divided by 3.0)
```

Figure 8.3: Pseudo-code of the radial constraint operator

---

Overall the application of this operator will cause all CDS elements to be progressively moved to within the relevant *radial\_constraint* pertinent to the relationship between each discrete CDS element and the gateway element; this is represented by the bounded spherical surface shown in figure 8.1. Once an element has been positionally mutated such that it is within the *radial\_constraint*, (Stage 3 in figure 8.3), all subsequent positional mutations will, in general, preserve this ideal distance magnitude.

There are three ways that an element can be moved out of this ideal constraint condition. Firstly, if a mutation event causes one of the dependent Cartesian coordinates to impinge on a boundary, then the constraint condition is not likely to be met as the boundary value is unlikely to be the required value to satisfy the constraint calculation. The second will follow the application of the crossover operator, this is likely to result in a number of elements being given Cartesian axis settings that will effectively *move* them quite dramatically; this could place them outside of the ideal distance, but equally it could move the elements closer. The third way is if the low probability *jump* or *swap* operators introduced in section 7.5 are set active. Whilst these interventions could be seen as drawbacks they have been left in the system for simplicity (and processing speed), but most importantly because they perform necessary randomising effects of providing a means of ensuring a continuing level of diversity within the population.

As depicted in figure 8.3 the GCO functions by modifying two, randomly selected, axes as a pair, the remaining axis is then sufficiently modified to either finally resolve the *radial\_constraint* or to get significantly closer to it. It is also possible to modify all three axes simultaneously by using a  $[3 * 3]$  matrix to represent the X, Y & Z axes. This will work correctly for much of the time; however it is not possible to guarantee that the 3-dimensional orthogonality of the X, Y & Z axes will be maintained under all conditions using this simple matrix approach.

The general Quaternion matrix method has been the subject of many research projects including those of (Chi, 1998) [17], (Hart et al, 1994) [55], (Vicci, 2001) [127], (Kuipers, 2000) [78], (Holin, 1999, 2003) [60; 61], and (Wheeler & Ikeuchi, 1995) [136]. This method could have been used as it does not suffer from the 3-dimensional orthogonality limitation, this is probably the ideal solution, especially as it can also provide the means by which a number of CDS elements could be easily mutated, rotated or otherwise translated and still remain in the same relative geometry, (and therefore maintain their local fitness). Some development work was undertaken to make use of this technique for another type of heuristic operator (usage of this type of operator is partially described in the concluding chapter 9, but limited time availability prevented the completion of the basic design. The active mutation method described above was the original one developed for this operator, it does maintain the orthogonal relationship between the axes, and also allows the use of existing functions used by other similar mutation operators.

### 8.5 Constraint constraint operator: EA simulation results

The plotted averaged and normalised *best-fitness* results of the GCO simulation are shown in figure 8.4 and summarised in table 8.1. The initial gradient of the plotted data indicates a very strong level of convergence towards the ideal of minimum overall fitness for all problem sizes, followed by an equally strong turning point indicating the point where the operator stopped being effective. The smaller problem sizes, PS10 to PS50 are solved to a reasonable level by the operator, the larger problem sizes, PS100 & PS200, show a distinct levelling out, in the PS200 case it is almost a flat-line with little evident convergence. Comparison with the *Standard EA* later in the section will demonstrate if the terminal level of fitness achieved is in any way superior.

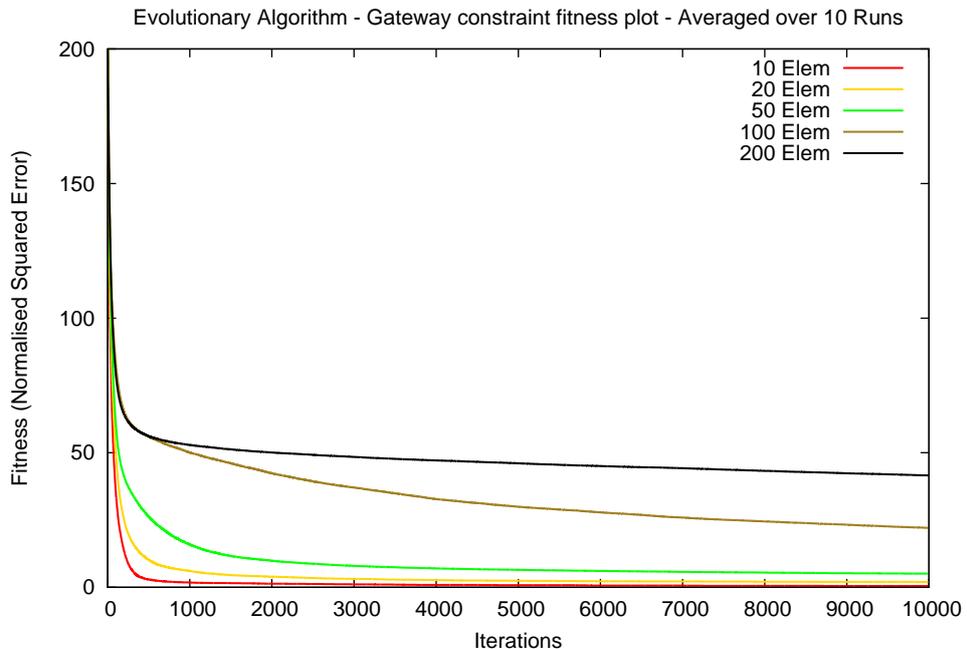


Figure 8.4: Gateway constrained EA fitness plot

One conclusion that can be drawn is from the general shape of the plot, with the sharp turn in the convergence trend being of particular interest, the restrictions imposed by the GCO appear, ultimately too severe; an alternative view of this may be that the operator is not sufficiently *intelligent* enough when it comes to continual mutation when restricted by a bounded situation. The latter point has some justification as the overall design philosophy has been to maintain the random approach to mutation as much as is possible; this is a theme that will be re-visited in the conclusions chapter.

Problem size	PS10	PS20	PS50	PS100	PS200
<b>All-run starting fitness condition</b>					
Initialised fitness	557.260	324.363	279.141	299.746	234.464
<b>Terminal fitness</b>					
Absolute Best	0.1094	0.5223	2.793	15.749	31.885
All-run Mean	0.527	1.454	6.207	26.106	39.668
All-run Worst	3.647	6.607	11.195	34.259	48.845
Average Delta	556.732	322.90	272.933	273.639	194.795
Average Delta %	99.905	99.551	97.776	91.290	83.08
<b>Range Groups</b>					
Group definition	[10,D]; 0≤:<2; 2≤:<5; 5≤:<10; 10≤:<20; 20≤:<100; 100≤:;				
Weighting factors	10.0 1.0 0.1 0.01 0.001 0.0001				
Hits metric	9 1 0 0 0 0	9 0 1 0 0 0	0 5 3 2 0 0	0 0 0 2 8 0	0 0 0 0 10 0
Integration value	31528.60	46262.82	103254.82	819468.56	1050340.14
Terminal value	0.01098	0.011	0.187	35.714	100.0

Table 8.1: Gateway constraint operator EA results table

The data in the table 8.1 illustrates the dichotomy in the results that clearly exists between the two groups as described above (PS10 to PS50 and PS100 to PS200). There is a clear, and significant *drop-off* in the **Terminal fitness** set of results that is matched by an equally noticeable degradation in the **Range Groups** part of the table. In particular the *Integration value* between PS50 & PS100 are subject to an eight-fold increase, equally the difference between the PS100 & PS200 is a twelve-fold increase. This indicates quite clearly that the overall performance of the GCO is markedly inferior with the larger data-sets.

The perennial problem for many EA based simulations is how to maintain a sufficiently strong level of diversity on the population, the effect of the GCO, whilst initially very strong, ultimately plays against it; this is very obviously depicted in the fitness difference plot shown in figure 8.5, the zero-line data is defined by the results of the *Standard EA* simulation from chapter 7. This can be clearly seen whenever the plot lines enter the positive zone; which indicates that the GCO has produced worse fitness results compared to the *Standard EA*.

As can be seen from this plot the performance of the GCO is noticeably superior to the *Standard EA* up to approximately 2500 to 4000 iterations, depending on the problem size. This represents an encouraging result that, on its own, does not provide the best performance overall; again, as with many aspects of EA, the level to which an operator is applied needs to be understood as fully as possible for the best performance.

## 8. HEURISTIC AND HYBRID OPERATORS

---

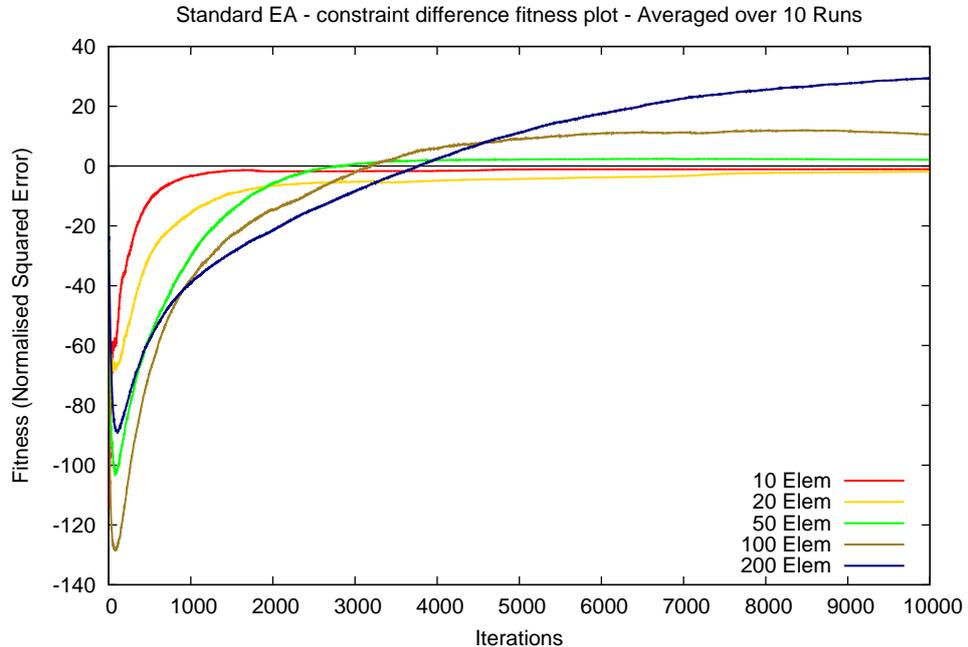


Figure 8.5: Standard EA (base-line)/gateway constrained fitness difference plot

### 8.6 Hybrid operator simulations

The previous section clearly demonstrated that the *Gateway Constraint Operator* (GCO) has a positive effect on convergence under certain conditions. In this section the objective is to further investigate some alternative ways to use the GCO to the best advantage. The *Standard EA mutation operators* are quite effective when used as the sole type of mutation operator. The logical conclusion from this would be to combine the two different operator types in a hybrid simulation in order to establish if the strong points of both types can produce an overall improvement in convergence and terminal fitness.

There are two initialisation file settings that are used to control the hybrid operator switch over point, this can be set at a fixed iteration or at an iteration value that is scaled according to the problem size (*exp\_OpHybridSwitchMoteFactor*). The latter is the preferred method as this allows the overall problem size to be taken into account before the automatic switch parameter (*exp\_OpHybridMaxSwitchVal*) causes the switch, this has been empirically set to an appropriate value. The active settings are as shown in figure 8.6.

---

Parameter	Setting
<b>exp_OpHybridMaxSwitchVal</b>	1000 Maximum operator switch iteration
<b>exp_OpHybridSwitchMoteFactor</b>	10 Switch at n * element count (100, 200, 500, 1000, <b>2000</b> ) *

Notes.

\* exp\_OpHybridMaxSwitchVal ensures 2000 iteration switch point is never reached

Figure 8.6: Initialisation file: operator switch parameters

### 8.6.1 Standard Gateway Constraint Operator hybrid results

There are two simulations in this section that are nearly identical, the exception being in the order that the operators are applied, both switch at the same point as controlled by the settings given in figure 8.6. The results are presented in two stages, the first stage consists of two fitness plots and a combined data table.

The second stage compares the two hybrid simulations directly in the fitness difference mode, finally the hybrid simulation that showed the best performance is compared with the *Standard EA* simulation from chapter 7 used above in figure 8.5.

The first simulation uses the *Standard EA mutation operator* as the initial operator, the fitness results plot for this are shown in 8.7. The second simulation starts with the *Gateway Constraint Operator*, the fitness plot is shown in figure 8.8.

## 8. HEURISTIC AND HYBRID OPERATORS

---

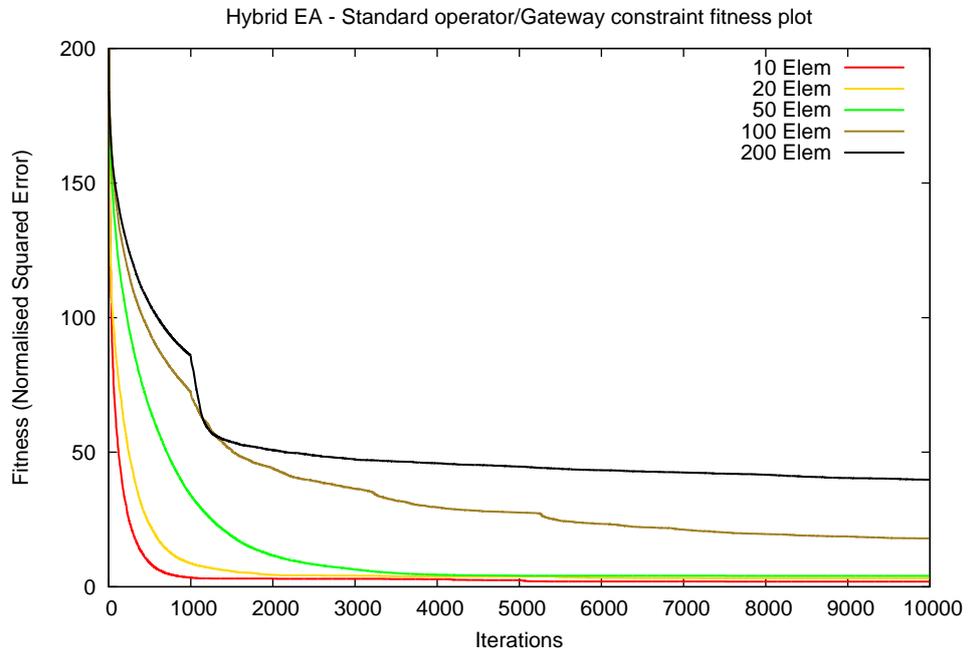


Figure 8.7: Standard EA - gateway constrained operator (switched) fitness plot

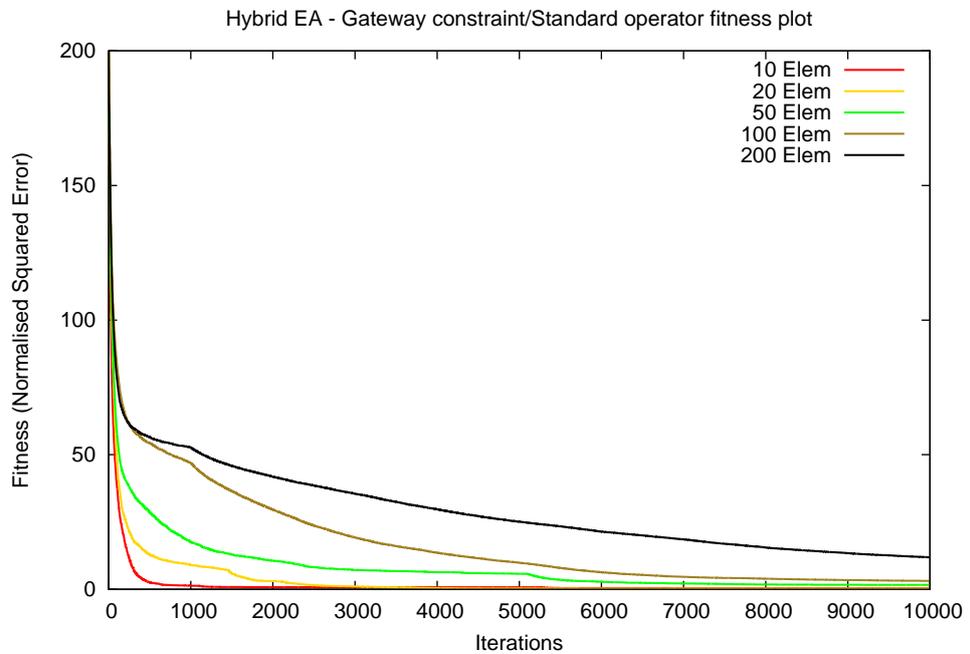


Figure 8.8: Gateway constrained - standard EA operator (switched) fitness plot

The results plots are similar in that they both show a quite good level of convergence across the problem sizes. The operator switch over point is quite visible as the *kink* in the plot-line around 1000 iterations on some of the problem size plot-lines.

There are also instances that occur after the 1000 iteration point where a similar *kink* is visible, it is not entirely certain as to the cause of these, but is felt that they are as a result of a quirk of the GCO (they do not appear in any other simulation type), and not evidence of an additional operator switch; once the operator switch has occurred it does not switch back again. One possible explanation of this *quirk* is that this is a direct result of limiting effect that the GCO may have of the general population diversity as discussed above. The more generally random *Standard EA* mutation operators can be expected to improve the diversity and that this causes the additional *kinks*, in effect these are noticeable improvements in overall fitness.

The first plot (8.7) that starts with the *Standard EA* mutation operators shows clearly inferior results to that in the second plot (8.8). The inferiority is in all aspects of the run-time performance, including the initial gradient, the general fitness level at the switch over point and the terminal fitness of the PS100 & PS200 problems sizes.

<b>Problem size</b>	PS10	PS20	PS50	PS100	PS200
<b>All-run starting fitness condition</b>					
Initialised fitness	557.260	324.363	279.141	299.746	234.464
<b>Standard EA/Gateway constraint operator</b>					
<b>Terminal fitness</b>					
Absolute Best	0.148	0.404	1.156	10.024	29.384
All-run Mean	1.933	3.188	4.069	17.967	39.773
All-run Worst	5.474	7.023	15.688	29.818	49.317
Average Delta	555.32	321.17	275.07	281.77	194.69
Average Delta %	99.653	99.017	98.542	94.006	83.036
<b>Gateway constraint/Standard EA operator</b>					
<b>Terminal fitness</b>					
Absolute Best	0.457	4.250	3.629	5.234	13.313
All-run Mean	0.403	0.225	1.607	3.108	11.857
All-run Worst	3.287	0.284	4.138	6.123	25.115
Average Delta	556.85	324.13	277.53	296.637	222.60
Average Delta %	99.9278	99.9307	99.4243	98.9629	94.9427

Table 8.2: Operator (ordered) results table

## 8. HEURISTIC AND HYBRID OPERATORS

---

Problem size	PS10	PS20	PS50	PS100	PS200
<b>Range Groups</b>					
Group definition	[10,D]; 0≤:<2; 2≤:<5; 5≤:<10; 10≤:<20; 20≤:<100; 100≤:;				
Weighting factors	10.0 1.0 0.1 0.01 0.001 0.0001				
<b>Order: Gateway constraint/Standard EA operator</b>					
Hits metric	9 1 0 0 0 0	10 0 0 0 0 0	8 2 0 0 0 0	3 5 2 0 0 0	0 0 5 3 2 0
Integration value	30164.23	48462.18	104120.70	278602.38	569119.36
Terminal value	0.01098	0.0	0.012195	0.028409	1.8797
<b>Order: Standard EA/Gateway constraint operator</b>					
Hits metric	5 4 1 0 0 0	5 2 3 0 0 0	3 5 1 1 0 0	0 0 0 6 4 0	0 0 0 0 10 0
Integration value	43077.612	84853.51	318633.88	747324.16	1502977.60
Terminal value	0.0184	0.0191	0.0284	15.62	100.0

Table 8.3: Operator (ordered) range group results table

The general superiority of the simulation that starts with the *Gateway Constraint Operator* is confirmed by the tabulated data in tables 8.2 and 8.3. In particular the **Terminal fitness** data in table 8.2 clearly demonstrates that this simulation type achieves a considerably better fitness level across all problem sizes. This is confirmed by the **Range group** set of results with the **Integration value** and **Terminal value** data showing good results; with even the PS200 problem size having a **Terminal value** that is close to the ideal of zero, (this was achieved by PS20). The **Integration value** data for all problem sizes do still show that there is still some way to go before a perfect result is to be achieved.

### 8.6.2 Hybrid operator and standard EA comparative results

This final pair of plots are presented here as the final confirmation of the effectiveness of the hybrid simulation approach. The plot in figure 8.9 is the iterative difference plot between the two hybrid simulation types, the zero-line is represented by the hybrid that starts with the *standard* mutation operator. Therefore, as demonstrated above, anything below the zero-line indicates that the non=zero-line data is superior; in this case the hybrid that started with the *Gateway Constraint Operator* has obviously demonstrated the stronger overall performance. There are a couple of areas where the *standard* mutation operator hybrid is in positive territory but these are not greatly significant and the terminal fitness is in favour of the hybrid that starts with the *Gateway Constraint Operator*.

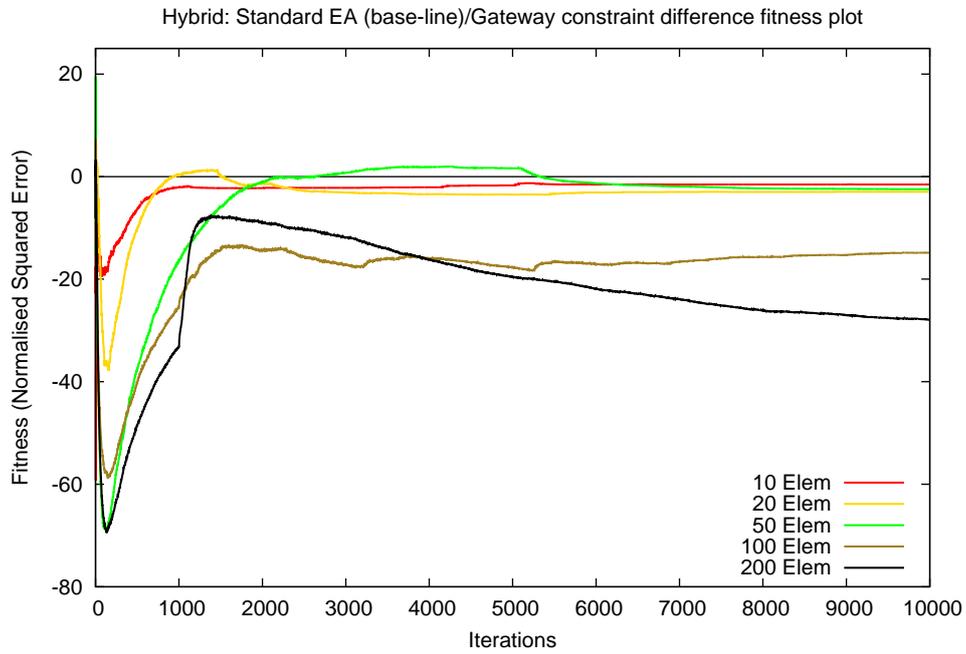


Figure 8.9: Standard EA (base-line)- Gateway Constrained operator Fitness difference plot

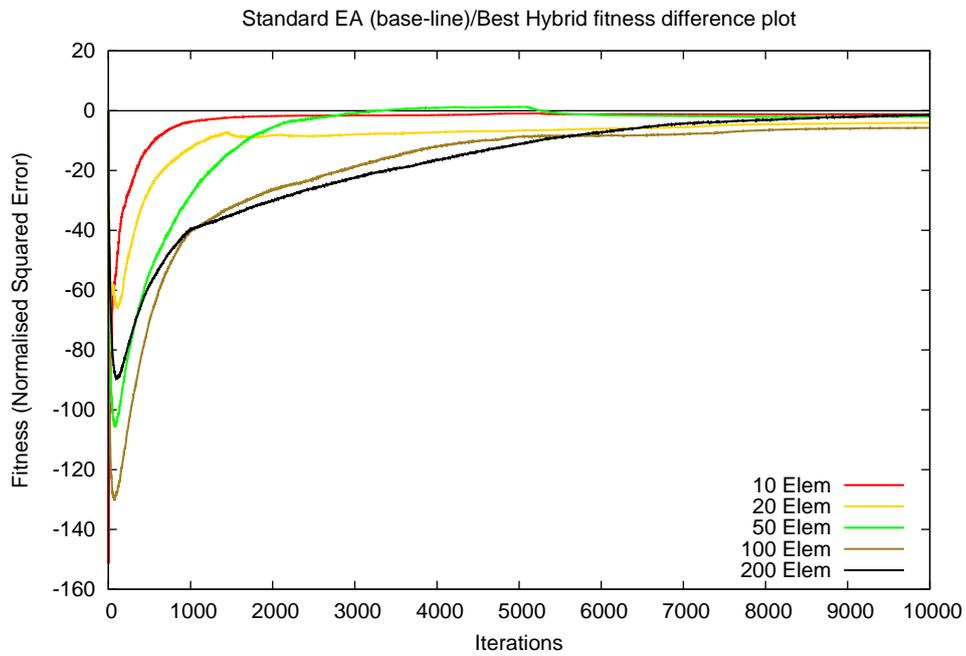


Figure 8.10: Standard EA (base-line)- best hybrid operator Fitness difference plot

## 8. HEURISTIC AND HYBRID OPERATORS

---

The second plot included in this section is shown in figure 8.10, this is also a comparative difference plot. In this instance the zero-line data is provided by the *Standard EA* simulation from chapter 5 as previously in this chapter. The objective is to confirm, visually, if the best hybrid simulation as defined in figure 8.9 is better overall than the non-heuristically controlled standard EA. As can be seen in the plot nearly all of the plot-lines are below the zero-line indicating that the overall performance of the best hybrid simulation does produce superior results.

### 8.7 Chapter summary

The underlying premise of the work of this chapter has been to illustrate the process by which the determination of some search space derived characteristics may be used to reduce the number of possible candidate solutions that could exist within that search space. This may be achieved by discarding, (or not considering), those candidate solutions that are considered as being ‘*impossible permutations*’ as they do not meet some pre-determined constraint(s).

The results of the simulations clearly indicate that reducing the search space in this manner demonstrates the existence of some usefully beneficial improvements to some aspects of the convergence profile in the initial stages of the run. There is also ample evidence that over-reliance on such techniques are likely to increase the probability of effects that are not beneficial to the terminal objective fitness. There is ample evidence in the plotted results that over-reliance on heuristic operators can have a negative impact on the overall diversity of the chromosome population. Such a situation is almost guaranteed to have an equally deleterious effect on the overall and terminal objective fitness, especially of the more complex problem sizes.

Clearly there is a need to minimise any such negative impacts by the avoidance of, or compensating for, any excessive evolutionary pressure to allow the process of evolving candidate solutions to proceed at least as well as if the heuristic operators had not been applied. Although there were some existing methods in the model software to provide some means of restoring some level of population diversity, these proved to be less than effective in compensating for the effects of the Gateway Constraint operator.

The addition of the hybrid mode algorithm type provided evidence of one type of mitigation strategy that still allowed the positive benefits of the search space reduction approach in one stage, whilst correcting for the population diversity restriction in another stage. The ordering of operator application, or heuristic, types was clearly demonstrated to exhibit significant influence on the overall performance of the algorithm. A general statement to summarise this point might be, to use the operator that provides the strongest search space reduction approach heuristic to gain the best possible benefit in terms of the rate of initial convergence and then to switch to an operator with a less restrictive function to compensate for any population diversity restriction.

---

This approach also circumvents the additional possibility that a restrictive operator ‘*may*’, inadvertently, prevent certain strong candidate solutions from ever being generated during the run-time of an algorithm that makes use of them.

In final summary, the primary intention of this process is to improve the probability of generating a set of suitably fit candidate solutions, the secondary outcome of this approach may also be a useful ‘*speed-up*’, in terms of processing time. The latter should not to be trivialised, as with ever improving processing power there is a possible tendency to rely on the ‘*number-crunching*’ power available to compensate for an algorithm that could instead be made more ‘*intelligent*’ through some appropriate heuristic methods. Those introduced here are relatively simple, demonstrating that the process does not have to be overly complex, and yet still provide a useful level of improvement to the overall level of terminal objective fitness, and ideally a reduction in the run-time required to achieve them.

## 8. HEURISTIC AND HYBRID OPERATORS

---

**Part V**

**Conclusions**



## Chapter 9

# Conclusions and further work

In this chapter the contributions of the work presented throughout the thesis is reviewed followed by a discussion on some possible areas of future work. The first section is a simple discussion of the problem space in terms of the need to understand and combat the effects of global climate change. The effects of this are still theoretical to some extent, although there is a growing body of empirical evidence that directly indicates the effects are real and very current.

The second section covers the conclusions gained from the practical real-world modelled representation developed in the thesis. The first part continues the contextual description through the introduction of some of the practical aspects inherent in the co-location of objects in 3-dimensional space. The second part draws the conclusions from the design and implementation of the software system, and finally the results of the algorithms as applied are examined.

The final section is concerned with the future research work considered necessary if the system is to move from a pure software model to the realms of a real-world practical prototype system. This takes the form of two distinct sub-sections, the first describes some of the work directly inspired by the work described in the thesis that has already been started. The second part of this section covers the work considered necessary to enhance the representation contained in the model by improving the suppositions inherent most specifically in the quality of the input data-set.

### 9.1 Conclusions

The problem space and area of research can be considered as covering a wide area of subjects that must be considered and subsequently drawn together to form any sort of conclusion. Due to this complexity some boundaries to what may be considered will be inevitable, the overall picture covers the environment, global climate change, existing environmental monitoring systems and techniques, modern electronics, applied optimisation algorithms, and, finally software modelling systems.

## 9. CONCLUSIONS AND FURTHER WORK

---

The conclusions thus drawn must be set against this group of subjects as providing the contextual background, it should be noted that not all of these aspects will be directly relevant to all parts of the work of the thesis.

### 9.1.1 Global climate change

The major concerns of the present day world are largely centred around the continued evaluation and protection of the natural environment. The world-wide concerns with global climate change has broad support across many governments and research establishments, perhaps the most significant among the latter are the UK Meteorological Office, [119], the Intergovernmental Panel on Climate Change [66], the United Nations Framework Convention on Climate Change [122], and the National Oceanic and Atmospheric Administration [87]. Although there are many news reports of apparently significant alterations in world-wide short term weather patterns that frequently result in extreme local flooding, drought, or storm conditions, it is in longer term climate change that the real concern lies as this represents a real change to the long term equilibrium of the environmental system that cannot be determined by short-term variations alone.

At the time of publication causing most concern is the depletion of the polar ice-caps through the work of the Catlin Arctic Survey [113], and that of NASA [50], they are considered to directly reflect the changes in the constituents of the upper-atmosphere that directly affect the level of heating in the overall terrestrial atmosphere. As a result research into this has been extensive and, significantly, has world-wide governmental support.

This forms a significant part of the real-world background context to the need for research into improving both the quantity and types of data available as part of the ongoing efforts into understanding the causes and possible solutions to global climate change.

## 9.2 Conclusions on the work of the thesis

The objective of this section is to examine the work of the thesis in the development of the real-world modelled representation. This section is divided into discrete three sub-sections each of which differ sufficiently so as to require a different approach to the contextual background description.

### 9.2.1 Part I: Introduction, literature review and background

This part of the thesis is largely concerned with setting the background context, introducing the concept and a review of a sub-set of the existing literature appropriate to the overall theme. This part consists of the **Introduction, Literature review &**

---

**Background** chapters (3, 2 & 4) respectively.

The introduction chapter provided a simple high-level overview of the concept of the existing Radio Sonde system that produces a near real-time synopsis of the upper-air meteorological parameters as part of an overall strategy to provide everyday weather forecasting and, potentially of greater significance for the work of the thesis, in the field of global climate research. This chapter also introduced the concept of Smart Dust and how they may be used to imitate the Radio Sonde, and subsequently improve upon it by the development of a multiple element Smart Dust based Sonde system that can operate in 3-dimensions.

The literature review chapter considered a number of subjects that are all relevant to the work of this thesis. The subjects reviewed are a sub-set selected from the field of research work currently available, the latter is currently large and varied and thus required careful consideration so as to include only those felt most appropriate to this work.

These are listed below.

- Environmental monitoring using Smart Dust
- Localisation and Radio Signal Strength Indication
- An overview of ant colony and swarm optimisation
- Optimisation algorithms based on the evolutionary method

The conclusion from the literature review can be summarised thus, the limitations of the input RSSI data-set type are well recognised and understood, but that there is currently no consistent method of completely mitigating the effects to make the parameter reliable enough for general use. The second conclusion has to be that some form of optimisation algorithm is a necessity when trying to resolve a generalised co-location problem, as many research programmes have confirmed.

The overall objective of this background chapter identified and examined the most significant aspects of the many that form the existing real-world Radio Sonde system and those that are required for the augmentations of the Smart Dust multiple element equivalent to function correctly.

## 9. CONCLUSIONS AND FURTHER WORK

---

The areas discussed.

- Defining localisation
- Relative distance identification
- Evolutionary algorithms
- A general overview of range calculation
- **R**adio **S**ignal **S**trength **I**ndication (**RSSI**)
- An overview of the generic Radio Sonde monitoring system
- An advanced Radio Sonde
- The object oriented design paradigm

Figure 9.1: Real world abstraction

To assist in answering the question required an examination of these aspects in some detail, in the theoretical context of a multiple element Radio Sonde device operating in 3-dimensions.

It was concluded that all of the aspects listed in 9.1 are both necessary in their discrete subject areas and as part of the whole; specifically that they all represent concepts that are quite feasible to quantify in terms of the distinct and definitive parameters as discrete subject areas, and in their individual roles when combined into the overall real-world problem space abstraction.

### 9.2.2 Part II: The problem description

This part of the thesis addressed the design of the software model in the **Problem Description** chapter (4). This chapter is effectively in two distinct sections, the first describes and identifies the key aspects of the real-world upper-airspace abstraction in both hardware and software terms. The second aspect uses the information contained in this abstraction and transforms it into a high-level software design model from which the run-time code has been derived.

For the first section the primary objective is the determination of whether the Multi-element Clustered Drop Sonde (MCDS) is a feasible concept using the suggested Smart Dust (SD) hardware, specifically for this thesis the hardware and software system as supplied by Crossbow incorporated.

To achieve this it was necessary to examine a full run-time cycle in detail that would

---

define all aspects of the operation of such a system. This was pursued in the context of detailed knowledge of the low-level requirements and limitations imposed by the hardware and software that defines the Crossbow SD system. A further contextual level was imposed by the software language (C & C++), in this case the limitations were minimal as the object oriented methodology supported by C++ actually provided solid guidance on how the run-time requirements could be directly supported, and even augmented.

This process also required that the real-world environment in which an MCDS would be deployed would itself impose considerable restraints on its operation. This directly related to the fundamental aspect that forms the input data-set, this is constructed from the RSSI data generated by the run-time duplex wireless traffic.

The use of RSSI, even if it were a perfect representation of the distance magnitude, will always be a flawed method. Principally this is due to the variations imposed upon it by the wireless transmitters and receiver variations and the transmission medium itself. Some of these effects can be mitigated through design, others will always remain a random variance that cannot be allowed for exactly precisely because of the random effects. One of more significant deleterious effects that can be completely corrected is that of transmission clashes, this can be eliminated by the simple expedient of controlling the sequence when each discrete element is allowed to transmit. This *'round-robin'* approach will be effective in this, it does however introduce another drawback in that it will impose a minimum time required for all elements to transmit in any given epoch; the corollary therefore is that during that time the elements are likely to have moved slightly, thereby applying a further smearing effect to the relative positions within the geometry. The conclusion here is that RSSI imposes severe limitations on the ultimate accuracy of any reconstructed geometry such that it may be impossible to produce a completely accurate representation. Further to this the *'fuzziness'* thus implied makes the case for an optimisation algorithm approach a stronger one, especially if some heuristics can be applied to *'intelligently'* correct for these implicit errors.

The conclusion from this exercise was that the design strategy of the MCDS process would result in a robust system suitable for the task of software implementation. With the benefit of hindsight this proved to be correct as the original design concept remained largely unchanged with the only alterations occurring as a result of the code design that itself suggested some subtle improvements to the overall concept.

The second section of the chapter is largely concerned with the further design and full implementation of the software model. This stage actually led to the full realisation of the concept of the optimisation algorithm generalisation framework in the form of the Virtual Base-class Framework (VBF). It would have been easy to implement a single piece of software to model the 3-dimensional environment in which the elements

## 9. CONCLUSIONS AND FURTHER WORK

---

of the MCDS could be represented, <sup>1</sup>.

The VBF approach was to identify all salient aspects that define a generalised optimisation algorithm, (not just an Evolutionary Algorithm), and to transform this information into a design specification. The point of this is to provide a mechanism that supports the implementation of new algorithm simulations, (for example through different mutation and crossover operators, or objective functions) to operate, in effect, as derived class hierarchy plug-ins. This methodology, directly supported by the principles of object oriented design produce a software system that had extensive code module re-use that enforced the ‘*test-once*’ philosophy that greatly improved the reliability of the software.

The conclusion must be that this the approach was completely successful as evidenced by the same basic class hierarchy (the VBF) being used for all simulations used in the thesis from the purely random to the applied heuristic algorithms.

### 9.2.3 Part III: the ‘*standard*’ algorithm simulations

The objective of this part was to demonstrate the operation of the 3-dimensional geometric reconstruction software model built directly from the Virtual Base-class Framework (VBF). The ‘*Standard*’ Algorithm Simulations part of the thesis is in three distinct sections that broadly reflect the three separate chapters in this part of the thesis, specifically chapters (5 & 6).

The first section is the ‘*run-time base-line using non-optimised algorithms*’, this detailed the implementation of two random based simulation algorithms that are so simple in operation that the term ‘*algorithm*’ is almost superfluous. This is an intentional situation as the primary objective was to demonstrate that the VBF itself does not, and cannot, impose any evolutionary pressure on any convergent behaviour that might be present in the results.

There were two random algorithms used that were broadly similar, but with one major difference. These were the ‘*Random Landscape Search*’ and the ‘*Random Progression Search*’ algorithms. These used the same input data-set, over the same number of iterations, the objective was to see if they could find a suitably fit candidate solution by chance. The difference between them was that the ‘*Random Landscape Search*’ mode reset back to the starting condition after each iteration to ensure each solution had a common start point and therefore the same chance of being perfectly fit. By not applying the iterative reset step the ‘*Random Progression Search*’ always built each new candidate solution using the geometric state as generated during the previous iteration, thus ensuring that a level of ‘*memory*’ was retained during the simulation.

---

<sup>1</sup>this was actually done in the form of the input data-set generator required to build some example geometric data

---

By ensuring that the details of the algorithms used are as maximally random as possible with minimal evolutionary direction applied at any stage will reveal any convergent behaviour that cannot be assigned to the algorithm itself. This would leave only one possibility that the VBF based model was responsible.

The results of the two random algorithms were very different, the ‘*Random Landscape Search*’ failed to produce any convergent behaviour at all as it flat-lined and stayed that way, this result was expected. The ‘*Random Landscape Search*’ on the other hand exhibited a strong and unexpected level of convergent behaviour. This was limited however and flat-lined after approximately 1000 iterations with no improvement after that point. The probable reason for a result such as this is that the input data-sets have been deliberately set to be as ‘*unfit*’ as possible<sup>1</sup> this seems to have resulted in that virtually any form of selection will result in an improvement to the overall objective fitness; an aspect absent from the ‘*Random Landscape Search*’ algorithm.

The conclusion was that this rather unexpected behaviour could not be attributed to the VBF applying any default evolutionary pressure as convergence would also have been evident in the results for the ‘*Random Landscape Search*’.

The second section concerned the ‘*setting algorithm specific parameters*’, this refers directly to chapter 6, (Initial runs: setting algorithm specific parameters).

The parameter list is given below.

- Chromosome (number of alleles) mutation rate
- Chromosome population size
- Selection mode type
- Mutation operator type
- Crossover operator type
- Chromosome mutate/accept (unchanged into new population) ratio

The approach was to set up a series of run-time simulations to establish a standard group of settings for the basic run-time parameters normally associated with Evolutionary Algorithms (EA). The final parameter settings were all arrived at through empirical means with each discrete parameter being the sole subject of a dedicated EA simulation run with all other significant parameters set to reasonable default values. This process provides a best overall performance setting for all problem sizes (PS10 to PS200).

---

<sup>1</sup>This was achieved by creating the input data-set element cluster as widely dispersed as possible, and the model initial state as a tight cluster at one point

## 9. CONCLUSIONS AND FURTHER WORK

---

The results given in this part can be summarised thus. The software model exhibited robust and repeatable behaviour across all problem sizes, and for all parameters under investigation. This can be taken as confirmation that the concept behind the VBF model, for this problem search space is valid and reliable. Any anomalies in run-time operation would be very apparent in the results, apart from a few slight ‘quirks’ that can be attributed to the way that some discrete runs evolved this is felt to be a valid statement.

The conclusions drawn from this piece of the work is that the overall operation of the VBF as been confirmed, and that the software model as implemented is largely unaffected by the actual settings. Selection mode type being a particular example where the influence on run-time convergence was not particularly strong. Overall the results clearly indicate that the parameters can be set somewhere in quite a wide range without adversely affecting, the notable exception is that of the mutation rate where the ‘sweet spot’ was quite tight with a range of 3.0 to 10%.

The third section involved ‘*Optimisation algorithm simulations*’, this refers directly to chapter 7, (Optimisation algorithm simulations). In effect this chapter contains the first set of simulations that are using the complete set of parameters as set in chapter 6. Using these settings the focus moved from attempting to determine a ‘best set’ of parameters to the first genuine attempt to solve the 3-dimensional geometric reconstruction problem using the VBF derived software.

The chapter defined two optimisation algorithms, the ‘*Stochastic Hill-Climber*’ (SHC), and the ‘*Evolutionary Algorithm*’ (EA). These algorithms were both applied to the same input data-sets, and, where applicable, identical parameters as previously defined.

The results for the SHC were surprisingly strong with good, if relatively slow convergence across all problem sizes, with particularly strong performance for the smaller problem sizes (PS10 & PS20), with PS10 effectively solved. This level of performance was something of a surprise, although the terminal objective fitness values for the larger problem sizes indicates room for improvement. Although speed of run-time operation data was not included for any simulation used in the thesis observation of the SHC indicated that it was, primarily due to its simplicity, considerably faster (in terms of time taken per iteration) than the EA. This is a useful observation that may be of use if a hybrid algorithm was being considered.

The results for the EA, as might be expected, show a considerably stronger performance in both convergence and terminal objective fitness over that of the SHC. The performance is better for all problem sizes indicating that the EA type of optimisation algorithm is better suited to providing a more complete solution to the problem space than the SHC. The final EA results are those of the large data-sets (PS250 to PS500),

---

these were included not only for completeness but to illustrate that the problem can be scalable to larger problems, it also indicated the need to consider methods by which the run-time performance of the software model could be improved as these simulations took a considerable length of time to complete.

The overall conclusion to be drawn from this section is that the VBF derived software model represents a viable and effective research tool, that it can be improved upon is also in little doubt as the results clearly indicate there is room for improvement.

#### 9.2.4 Part IV: Search space reduction & Heuristic Simulations

This part of the thesis, specifically chapter 8, was primarily concerned with the introductory stages of the process to improve the underlying performance of the software model in terms of the objective fitness values obtained, (in both the rate and the terminal value), and in the run-time speed of the software model.

The method chosen to achieve the objective of this part of the research was to develop methods to create a general reduction in the size of the search space. By reducing the potential number of solutions that *'may'* be present in the search space it can be reasonably expected that the probability of generating a good set of candidate solutions should be improved. This should also result in a reduction in the time required to generate the set of candidate solutions, if the implementation of the search space reduction methods are designed in such a way as to ensure that the search process is not excessively impeded, and of greater significance, that the best candidate solutions are not unintentionally eliminated from the modified search space. To achieve this ideal state it is imperative that any heuristic method applied to search space reduction remains consistent with the *'ethos'* of that search space so as to remain both complementary and intrinsic.

The approach to solving the problem was by the development of operators that apply a search space specific heuristic that has the capability to reduce the number of potential candidate solutions in the search space, and by the development of hybrid algorithms that allow the run-time, dynamic switching of the heuristic operator with its *'standard'* counterpart.

The heuristic mutation & crossover operators applied a constraint on the possible 3-dimensional positions that a Smart Dust based Drop Sonde element could occupy by restricting them to a part spherical boundary, calculated using the RSSI data between each discrete element and the positionally fixed gateway element. This method used data that is intrinsic to the definition of the real-world abstraction of the search space, in so doing, (assuming that it is correctly designed *'and'* implemented), the resultant operators can be expected to maintain sufficient data integrity to not eliminate any of the set of *'correct'* candidate solutions.

## 9. CONCLUSIONS AND FURTHER WORK

---

It was found that this produced a significant increase in the initial stages of convergence that clearly indicated that the constraint applied was having the desired beneficial affect on the generation of candidate solutions. It was also noted that after the initial improvement in the rate of convergence the terminal objective fitness was less good across the larger problem sizes, (PS50, PS100 & PS200), the results for PS10 & PS20 showed an improvement across all iterations. The conclusion was that the drawback to the application of the gateway constraint heuristic was probably causing a reduction in genetic diversity in the chromosome population for the larger problem sizes, the easier problems were unaffected simply because they were easier.

One solution considered was to develop the hybrid algorithm that utilised the gateway constraint operator for the initial stages to gain the performance advantage that it offered and then to switch to the unconstrained, or standard, operators as a means of regaining some of the lost genetic diversity. Whilst this would mean that some of the 3-dimensional positions of the Drop Sonde elements would stray from the spherical boundary constraint the *'normal'* operation of the EA would very quickly evolve them out of the population and restore the terminal objective fitness values lost to reduction in genetic diversity.

It was observed that the best performance was obtained by, as anticipated, using the gateway constraint operator for the first stages of the run, and then by switching to the standard operators, thus reinstating a *'standard'* EA. The final conclusion was that this area of research has great potential for future developments, especially as it may provide an alternative approach to some aspects of multi-objective optimisation algorithms; specifically in the simplification of the objective fitness function by replacing some of the objectives by heuristics that are applied at run-time thus removing them from consideration, completely, or as in the hybrid simulation, as a temporary measure.

### 9.3 Further work

One unavoidable conclusion from the work of the thesis completed thus far is that there is the potential, and need, for more research in this subject area. The structure of the Virtual Base-class structure software model is quite adequate for the purposes, and has demonstrated ample flexibility to meet the majority of future research developments that can be currently envisaged. The model is currently, (and will always be to some extent), incomplete in terms of the modelled representation of an upper air-space containing freely distributed wireless connected elements. Ideally any future work will address the limitations nearly all models will exhibit by a combination of improving the detail of the modelled representation itself, the data used in the run-time input data-sets, and that which defines the model run-time environment parameters operating on the input data-sets.

---

### 9.3.1 The error model

The first priority therefore will be the identification, quantification and subsequent handling of RSSI derived input data-set errors in the model before it could be considered as being a complete representation of the upper-air ‘*Drop Sonde*’ problem space.

The sort of errors that can be expected in the RSSI data-set can be modelled quite successfully by the technique described in this section.

All of the simulation models have used an error free, *idealised*, RSSI input data-set as the source of the geometric configurations. Part of the reason for this is that there is no real-time RSSI data for a clustered ‘*Smart Dust*’ Drop Sonde, because, as far as is known at the time of publication, apart from the very early experiments performed for this work, such a system does not actually exist as an operational device.

The input data-sets have been randomly generated using the simulation model in reverse-mode as described in section 3.3.3 and appendix D. A large number of random data-sets were generated to help ensure that they would not be a cause of any bias in the results by merging (averaging) results from across a number of input data-sets. These data-sets were all generated as idealised and therefore totally error-free, they were also maintained throughout the work in a totally unchanged state to maintain consistency between simulations.

Some work on the effects of errors in the RSSI data-set was performed at an early stage in order to observe the effects on the performance of the early development version evolutionary algorithm model software (it was fairly primitive at the point in time). As the data-sets were idealised the errors were generated within the model according to a number of parameter settings in the *Error Fitness object params* part of the initialisation file in appendix C, they are reproduced in figure 9.2.

Parameter	Setting		
<b>exp_CorrectErrorOnceOnly</b>	<b>0</b>	(False)	Pre-process errors
<b>exp_DontAllowGatewayError</b>	<b>0</b>	(False)	Gateway assumed immovable
<b>exp_PktMissing_pc</b>	<b>20</b>	(0..60%)	RSSI missing data
<b>exp_PktLowPayload_pc</b>	<b>20</b>	(0..60%)	RSSI value is reduced distance
<b>exp_PktHighPayload_pc</b>	<b>10</b>	(0..60%)	RSSI value is increased distance
<b>exp_AllowMissingPktReverse</b>	<b>1</b>	(True)	Use reversed indices data

Figure 9.2: Initialisation file: Original input data-set error parameters

The effects of this simple error model were quite predictable, they were observed as causing degradation to the overall convergence and terminal fitness across all problem

## 9. CONCLUSIONS AND FURTHER WORK

---

sizes (at the time these were in the range PS5 to PS100). Details of this phase of the development and the subsequent results were not included in the thesis.

The work described above did reveal several aspects of the run-time performance that dictated the direction of the research from that point on. The first is that errors in the input data-set were shown, as was expected, to be a significant cause of convergence. The second was that the performance of the model could be significantly improved even with the idealised error condition;, with particular emphasis on the flexibility of the fitness function to handle the erroneous condition.

Finally, it was concluded that the characteristics of the problem space itself could be used to alleviate some of the deleterious effects of input data-set errors by mitigation. Specifically this led to investigation into the greater understanding of the problem space which subsequently led to the concept of simplification.

### 9.3.2 Field trials

Randomly modelling the variations in RSSI derived data is a necessary and important step in the development of the system to model the Drop Sonde cluster. As the ultimate goal is to produce a real-world Drop Sonde system it would be better still if *real* RSSI data from an active Smart Dust Drop Sonde cluster was available to use in the modelling stage. There have been a number of research projects that have attempted to characterise the RSSI value to distance relationship, including (Whitehouse et al, 2004) [94; 137], the drawback to these attempts is that it has been performed, either at ground level, or inside a large building. The conclusions from these trials was that the surroundings had an effect that was noticeable but not necessarily fully quantifiable. To attempt to alleviate some of the problems mentioned in the previous section it is proposed that the generation of *real* RSSI input data-sets would be greatly beneficial, in addition this would allow the simultaneous RSSI to distance characterisation to be performed.

The trial would provide the main source of raw RSSI data to form the basis of all subsequent work regarding the localisation/3-D modelling system. To develop a defined RSSI dataset, through empirical means that will characterise the basic relationship between transmitted signal power, received RSSI data and known (increasing) physical distance.

#### 9.3.2.1 RSSI field trials: Data-set verification

The purpose of the first stage in the trial will be to obtain the raw RSSI data required to determine the fundamental characteristics that define the use of RSSI in the real-world environment. This can be devolved into two main requirements, firstly, the rate at which the signal strength as *seen* by the receiving mote alters with increasing distance magnitude, and secondly, a measure of the random variability through the expected

---

variations in the performance of the wireless electronics and those imposed by environmental considerations. The RSSI data-sets generated through empirical measurement in this part of the trial will be transformed into one of the fundamental rules that will partly characterise the 3-dimensional configurations already encoded into the software modelling system, but not currently used in any operator. It is also possible that this data will form part of the code that individual Smart Dust elements may also carry in their normal runtime system.

### 9.3.2.2 RSSI field trials: Sub-shape signature generation

The next stage of the characterisation trial will be to take the raw RSSI data and create data structures that are self describing in terms of a specific 2/3-dimensional sub-shape. This involves suspending the mote cluster in predefined configurations, the sub-shapes listed below, in such a way as to minimise reflections and interference from the ground and nearby structures as described above. 2-dimensional

- Single Line (8 Points/elements in line)
- Flat Plane Shapes - Square/Pentagon/Hexagon/Heptagon/Octagon
- Parallel line (4 Points/elements in each line)
- Pseudo-Curve (n Points/elements)

3-dimensional

- Cube (8 Points/elements)
- Octahedron (6 Points/elements)
- Triangular Pyramid (4 Points/elements)
- Square Pyramid (5 Points/elements)

The RSSI values between individual pairs of node points in the pre-set sub-shape configurations will not add any further information to that which was gained in the first part of the trial. Each of the sub-shapes will generate an RSSI signature that will represent that particular shape as an abstraction that can be decoded using specific rules. The new information is contained within which will become apparent when the individual readings are treated as an integrated dataset with a defined internal structure. A simple example of a heuristic function is to use the raw data to form the basis of a prediction system that would generate realistic RSSI dataset (a sub-shape signature) for any required sub-shape. This data will also have a fundamental role to play in the calibration of this predictor system, the accuracy and reliability of the sub-shape configuration datasets will be of vital importance.

## 9. CONCLUSIONS AND FURTHER WORK

---

### 9.3.2.3 RSSI Field trials: The effects of antenna orientation

The sort of antenna design envisaged for the Smart Dust based Drop Sonde element will be omni-directional, the existing commercial devices use this type as part of the parachute that slows the descent. In practicality a totally pure omni-directional antenna does not exist, only one that closely approximates a point source; therefore work will be need to establish the effects of variance in relative transmitter/receiver orientation. Once the RSSI to distance magnitude has been characterised the effects of antenna orientation can be investigated. At short distances this is expected to be minimal as the *lobes* defining the transmission pattern effectively overlap, the field strength variations within these *lobes* is likely to be the overriding factor. For greater distances the gaps between the *lobes* is more likely to become a significant factor.

### 9.3.2.4 RSSI Field trials: Drop Sonde element '*flight*' trials

The final test, to *fly* the Smart Dust elements and to observe what actually happens. This would not be an easy task to perform with sufficient control of the relative positions of the CDS elements in flight in order to relate the physical geometry to the RSSI based geometry to establish how successful it was. As an example of the expected form this trial could take would be to tow a number of them behind a radio controlled aircraft and video them at the same time to provide some form of positional relationship information; an alternative would be to suspend them from one or two kites. Either way it would be an entertaining thing to do.

### 9.3.3 Search space reduction heuristics

The objective of this section is to introduce some methods of investigating how to improve the run-time performance of the geometric reconstruction model software. The factor common to all of the following sub-sections is that they all represent work that has been researched and implemented to some degree, but not sufficiently complete to warrant inclusion on the main body of the thesis.

### 9.3.4 Heuristic: Decomposition

In this section the initial part of the work examined how large problem spaces could be processed more efficiently, as well as providing one method by which the problem space could be simplified to make the process of mitigating the effects of errors easier to deal with.

The time taken to resolve a 3-dimensional geometry of clusters of CDS elements is largely dependent on the problem size, the order of this is an exponential function. It can be demonstrated therefore that the time taken can be reduced by the decomposition of the problem space into smaller sub-groups which are resolved in isolation and then recombined into a final set of candidate solutions.

---

The first stage in the process of simplification effectively implements the K-means technique to reduce the entire cluster into a number of sub-groups containing  $n$  elements, as described by research into Genetic K-means algorithms by (Krishna & Narasimha Murty) [77]. Of more specific interest to the work of this thesis are two research projects that examine K-means algorithm run-time conditions, the first by (Bradley & Fayyad, 1998) [11] investigates ‘*Initial Points for K-Means Clustering*’, and the second researched ‘*Constrained K-means Clustering with Background Knowledge*’ by (Wagstaff et al, 2001) [131].

The first method for this element of the research is the ‘*Gateway Constraint Operator*’ (GCO) described in chapter 8. This work, effectively groups two CDS elements together in a relationship based on retaining a set distance magnitude, whilst moving one around through mutation of the Cartesian axes. The natural progression of this is to extend the group concept to include a set number of CDS elements and treat them as sub-configurations within the overall problem space where there may be hundreds of CDS elements.

In terms of the run-time development this required a new set of mutation and crossover operators that recognised the sub-configuration groups and modified them accordingly. Space precludes a detailed description of this process in technical terms but the overall effect was to solve each sub-configuration group 3-dimensional configuration as a separate entity (and therefore the local fitness) and, to switch to normal (non) sub-configuration mode to solve the overall 3-dimensional geometry as represented by all CDS elements.

This work was completed to a large degree and was shown to function well, producing a reasonable improvement in the overall fitness, but more significantly in the time taken, (in terms of iterations), to reach a pre-defined fitness level.

### 9.3.5 Heuristic: Quaternion operators

The next phase of the heuristic methods in the development process of investigating how to deal effectively with the input data-set error condition was to extend the concept of sub-configuration groups described in the previous section. The drawback of the *decomposition* approach as described is that the sub-configuration relative distances are not maintained when after the switch to the *standard operators* has been performed, this is not necessarily a drawback as the overall stochastic operation of the EA is maintained, and excessively restrict CDS element movement can lead to problematic reductions in population diversity.

Nevertheless it was concluded that if the fitness of each sub-configuration group was sufficiently good then it made sense to maintain this fitness (through the relative positional geometry) and move all CDS elements together as a sub-configuration. During the development of the *Gateway Constraint Operator* (GCO), the use of matrices was

## 9. CONCLUSIONS AND FURTHER WORK

---

considered, and at that time, rejected as the existing operators could be used in a modified form.

Quaternion matrices have many uses in mathematics and software, and have been the subject for many research projects, examples of this include, amongst many others, the work by (Chi, 1998) [17], (Hart et al, 1994) [55], (Vicci, 2001) [127], (Kuipers, 2000) [78], (Holin, 1999, 2003) [60; 61], and (Wheeler & Ikeuchi, 1995) [136]. In optimisation algorithms Quaternion operators appear less prevalent, they are however commonly used in a number of software areas where the 3-dimensional displacement of points in space must be performed in such a way as to maintain the positional relationship, exactly what is required here. Example areas are in general graphics, universe representations, and in game programming. There are existing software libraries, such as Boost™ (initially written by Holin), and OpenGL™ [92], however it was decided to write the software myself in order to gain the greatest understanding of the process.

The basic development research on these Quaternion operators was very successful in that the experimental code allowed a number of points representing CDS elements to be mutated in the X, Y & Z axes and still to maintain their relative positions and, equally significantly, they maintain the relative orthogonality between the X, Y & Z axes; this is the main reason for using this type of matrix. The final implementation in a run-time optimisation simulation was not completed.

### 9.3.6 Algorithmic structure

Another area of software research was the development of algorithmic methods that are in the region of run-time structural control, that is how the overall algorithm progresses at a high-level as against the detail of the low-level operators. The idea behind this was twofold, the first that in a real-time situation the RSSI based data would be continuous and not just for a single time-step. The second was that information regarding the relative positions of CDS elements in the error case could be inferred from the error data gathered over several, recursive iterations.

#### 9.3.6.1 Algorithmic structure: Dynamic

The dynamic algorithmic structure was a relatively straightforward procedure as it had been designed into the overall expectation from the outset. The simulation switched to the next input file in the sequence as soon as the overall fitness was at a pre-determined level, or iteration point. The chromosome population as set for the terminal condition was maintained across the input files as the *seeding* condition for the next part of the convergence process. The results, (not reproduced here), clearly showed that the convergence continued at the *almost fit level* as set by the seeding effect of the terminal fitness for the previous input file. This work was completed successfully but was not included in the thesis, The initialisation file has provision for the use of a number of input files, each of which represents a time point.

---

These parameters are shown in figure 9.3.

Parameter	Setting
<b>exp_MultiInFile</b>	<b>1</b> (True) Multiple files
<b>exp_InFileTemplate</b>	<b>time_&lt;template_&gt;000 to &lt;template_&gt;999</b>
<b>exp_FileTemplateStartNum</b>	<b>0</b> First number in template
<b>exp_FileTemplateEndNum</b>	<b>10</b> Last number in template

Figure 9.3: Initialisation file: Multiple input file parameters

### 9.3.6.2 Algorithmic structure: Recursive

The recursive algorithmic structure was an experimental version of the dynamic algorithmic structure described above. The intention was to provide the means by which the status of the overall condition pertaining to the relative positional errors could be quantified. The expectation was that, in combination, the pre-processing of the input data-set, the overall function of the heuristic operators and general work on the algorithmic structure would represent a *snap-shot* of the overall situation; in order that further insights into the novel methods to reduce the effects of the input data-set errors. This work represents what was to be the final stage in providing a solution to the RSSI error model problem space, unfortunately time precluded its completion to any useful level.

## 9.4 Final conclusions

The work of this thesis addresses a real-world practical application in environmental monitoring, the realisation of which has been made considerably feasible due to the development of the Smart Dust™ system of generalised wireless enabled sensor platforms. That this commercial development has stimulated many new wireless sensor platform applications cannot realistically be in dispute, it has also been largely responsible for a significant number of new research programmes into an increasingly wide range of subject areas, including that of this thesis.

The need for research into the effects of global climate change has resulted in a strong world-wide determination to establish the facts of the situation, the result is a considerable increase in work into all aspects of environmental, meteorological and climatic monitoring. It can be expected that work in this field of research is therefore likely to continue at a fast pace, at least for the foreseeable future.

## 9. CONCLUSIONS AND FURTHER WORK

---

The final conclusion, in terms of this thesis, is that the work done thus far has indicated that the original concept of a 3-dimensional, multi-element upper-air meteorological monitoring system is sound, and that with further research could indeed result in a practical, powerful, flexible, low cost, real-world, 3-dimensional environmental monitoring system of the type envisaged in these pages.

# Appendices



## Appendix A

# Crossbow Technology<sup>TM</sup> Smart Dust system

All of the imagery and data tables reproduced here are copyright of Crossbow Technology Incorporated, unless otherwise stated.

The appendix has been divided into three sections:

1. The Smart Dust 'mote' elements.
2. The Smart Dust serial, USB and Ethernet interface devices.
3. The Intel XScale<sup>TM</sup> Stargate Ethernet computing device.
4. The TinyOS<sup>TM</sup> operating system and NesC<sup>TM</sup> programming language.

The original Crossbow Smart Dust devices, selected for the initial design abstraction have been superseded, such is the pace of development in this area of technology. There is still a vibrant research and development community using all aspects of the technology and this can be expected to continue. There are a number of other commercial companies and research groups that have also produced systems that can be categorised as 'Smart Dust'.

One significant operational difference of the recently developed systems is that they have now been designed to conform to the Zigbee specification (IEEE 802.15.4-2006). The Zigbee standard is a comprehensive specification covering many aspects of a wide range of modern technology, the very latest being 3-d television. The new aspect that is specifically relevant concerns the communication protocol, the specification defines a much higher wireless basic frequency of 2.4 GHz (compared to the original frequency range of between 315-916 MHz). As a generalisation an increase in the spot frequency will reduce the range for a given power density, especially if at low altitude such that

## A. CROSSBOW TECHNOLOGY™ SMART DUST SYSTEM

---

interaction with the ground will adversely affect the transmission-reception process. As Radio Sonde are not generally used at ground level it can be reasonably expected that this would have little negative effect on the range, although slightly more power would probably be necessary. A positive aspect of the new specification is that the replacement Chipcon™ radio 'chip' (the CC2400 - originally the CC1000) frequently used on the Smart Dust devices has been developed still further.

There are two significant advances of this development, the first is that the data transfer rate has been increased eight-fold over that of the CC1000, the second advance concerns how the RSSI and signal quality data is now generated. The RSSI data is still generated in the same way but this has been enhanced by sampling significantly more of the received transmission to give a more reliable figure, in addition a quality metric based on the received packet quality is available to supplement the RSSI. These combined metrics offer the prospect of a significant improvement to the overall accuracy of the RSSI to distance conversion.

In summary, the original Smart Dust was felt to be more than adequate for the projected task, the current systems are likely to offer significant functional improvements, even at the current development stage. This process of development can be expected to continue for some time yet, although it is unlikely that true dust mote sized Smart Dust would be necessary for this particular usage as a Radio Sonde. Other uses for 3-dimensional dynamic monitoring may well be required and these will still need a method for dynamic wireless generated geometric position determination.

### A.1 The Smart Dust elements

There are two Smart Dust motes from the original hardware set that both have a distinct functional role in the definition of the abstraction, this is explored further in a subsequent appendix. They are quite different in shape and size, and also as a direct result of this, in the number and type of interface ports available; however from a purely functional point of view they are identical.

#### A.1.1 The Crossbow MICA2DOT Smart Dust mote

This Smart Dust device has comparatively small dimensions and mass and would of the two types available be best suited as the processing element of the Radio Sonde in the abstraction of a functional sonde and communication system. The reduced set of inputs that result from this particular form-factor are still more than adequate for a simple sonde monitoring instrument suite of barometric pressure, relative humidity and air temperature. A very simple Radio Sonde high-level design is given in appendix B.

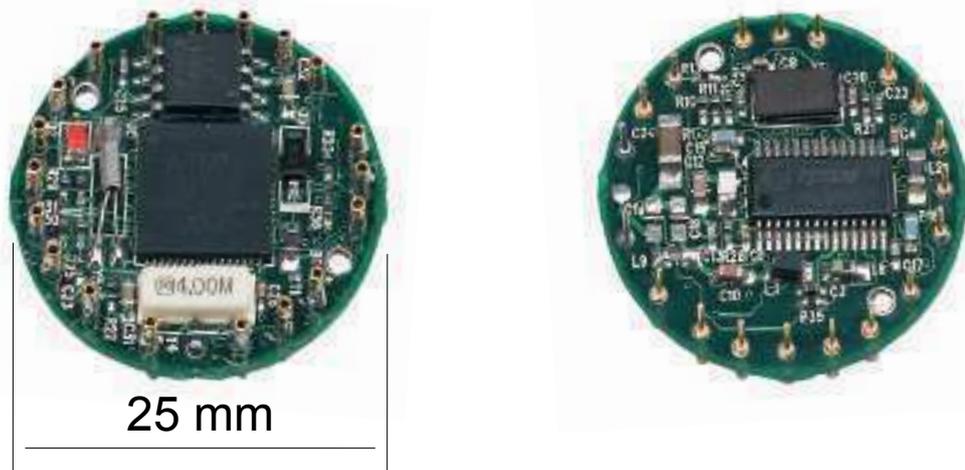


Figure A.1: MICA2DOT Smart Dust mote Circuit board view

The Mica2Dot device circuit board is shown above, to aid operation the mote is most often used with two other similarly sized components. The first is a passive interfacing circuit board fitted with a ring of connector sockets onto which the mote is mounted, the second houses a 3volt lithium 'coin-cell' battery. Neither of these are actually necessary for mote operation as for a bespoke installation, such as the projected sonde, hardware specifically designed the task will be used.

#### **A.1.2 The Crossbow MICA2/MICAz Smart Dust mote**

The Mica2 Smart Dust device is a more robustly constructed unit that, as a result, is more suited to forming the receiving station of the Radio Sonde communication system. When it is paired with an appropriate interface device (covered below) it provides the controlling wireless link between the Radio Sonde elements and the data storage facility prior to the post-processing necessary for even a standard Radio Sonde that would be tracked by a radar system.

### Technical summary of a MICA2(DOT) mote and sensor suite.

- CMOS miniature electronic construction
  - Around one million transistors in a small package (~mm<sup>2</sup>).
  - 128 Kbytes Programmable memory.
  - 512 Kbytes Data Storage Flash Memory.
  - Eight (six) 10-bit Digital to Analogue Converters.
  - Twelve (nine) Digital I/O channels.
  - Serial communication Universal Asynchronous Receiver Transmitter (UART).
- Duplex Communication System
  - Multi-Channel Radio (between 4 and 50 channels).
  - Data transfer Rate 38.4 Kilobits/second.
  - Programmable RF Power -20 to +5 dBm.
  - Outdoor Range, 500—1000 ft using a quarter-wave Wave dipole.
- Power
  - Current Draw,  $\approx 25$  mA (transmit)  $\approx 10$  mA (receive)  $< \approx 1$  uA (sleep).
  - 1—10 mW active, 1 mW passive at 1.0% duty cycle.
  - Primary battery (1 cm<sup>3</sup>) - nominally one year at 1.0% duty cycle.
  - Solar (10 mW/cm<sup>2</sup>, 0.1 mW indoors).
  - Vibration generator ( $\approx$ eW/gm).
- Micro-sensors
  - Acceleration, vibration, gyroscopic, tilt, magnetic, light, heat, motion.
  - Environmental, barometric pressure, temperature, humidity.
  - Chemical, detecting air borne agents (CO, CO<sub>2</sub>, radon, biological).
  - Micro-radar.
  - Micro-actuators (mirrors, motors, smart surfaces, micro-robots).

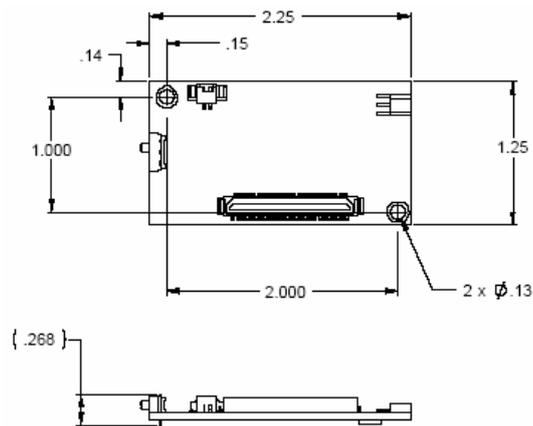


Figure A.2: MICA2 (z) Smart Dust mote (dimensions are in inches)

## A.2 The Smart Dust interface devices

The set of interface devices all have the primary task of enabling easy communication to the diminutive Smart Dust devices through dedicated hard-wired ports. The communication covers all aspects from data acquisition through to the downloading of the software programs into the devices in the first instance. Although the Smart Dust device can be programmed using the wireless link they still require an initial hard-wired programming step to enable this; there are potential security risks using this method and thus it is not recommended.

### A.2.1 MIB510 RS232 Serial Interface Board

The MIB510 interface board is a multi-purpose interface board used with Crossbow Smart Dust motes that use the ATMega128 microprocessor. The primary purpose of the MIB510 is to provide the communication bridge between an external computer and a Smart Dust mote for programming and data retrieval purposes. The bridge is effectively formed by a DB9 RS-232 serial port, in-system processor (ISP), and one of two connectors dedicated to specific types of mote. The RS-232 interface is a standard single channel bi-directional interface with a DB9 connector to interface to the external computer. The serial interface uses the transmit and receive lines only, and therefore has no hand-shaking or flow-control capability. The mote connectors allow for a MICAz (Zigbee), a MICA2 or a MICA2DOT mote to be connected.

The MICAz and MICA2 are both of the larger form-factor and are connected to the relevant connector on the top side of the MIB510. In addition there is a separate connector on the underside of the MIB510 that has a dual function, it can be used to attach a sensor board whilst a MICA2(z) mote is attached or it can be used to connect a MICA2DOT for reprogramming and data access.

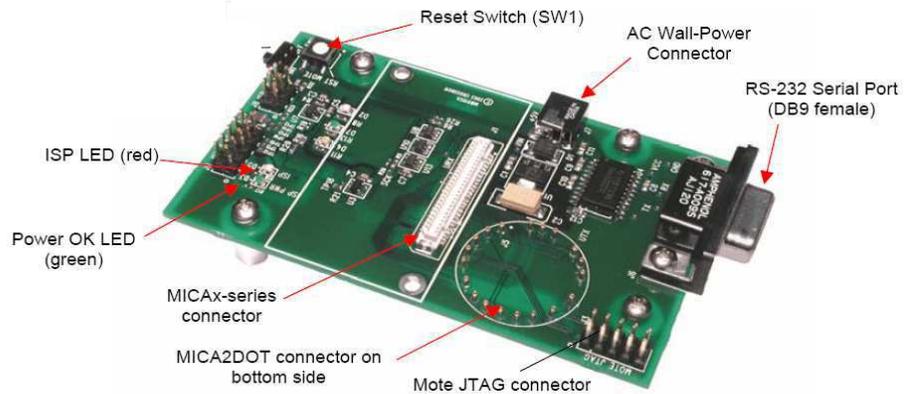


Figure A.3: MIB510 RS232 Serial Interface

The MIB510 uses an Atmega16L as the ISP to control communication and data transfer between the external computer and a connected mote. To program a mote the precompiled code is downloaded from the external computer through the RS-232 serial port and into the ISP, and then transferred to the mote.

---

### A.2.2 MIB520 Universal Serial Bus (USB Interface Board)

The MIB520™ provides USB connectivity to the MICA family of Motes for communication and in-system programming. It supplies power to the devices through USB bus. MIB520CB has a male connector while MIB520CA has female connector. It does not support the smaller Mica2Dot Mote and therefore is curiously limited.

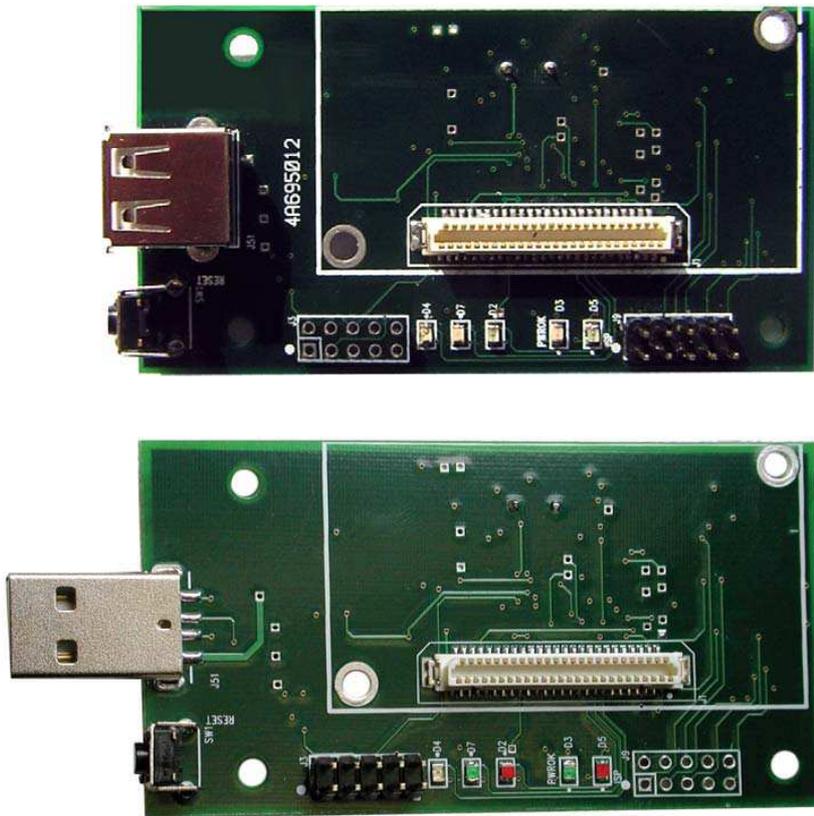


Figure A.4: MIB520 Universal Serial Bus Interface

### A.2.3 MIB600 Ethernet Gateway Interface Board

The MIB600 is functionally similar to the MIB510, however the addition of an Ethernet 10 Base-T LAN port makes it significantly more flexible. The MIB600 can access and program the motes in a very similar way to MIB510, with the notable exception that it is no longer possible to directly connect a MICA2DOT mote. The LAN port is key to the primary purpose of this interface device, whilst it can program motes it is most useful acting as a gateway between a deployed network of motes and a controlling and data logging computer that can be remotely sited anywhere in the world.

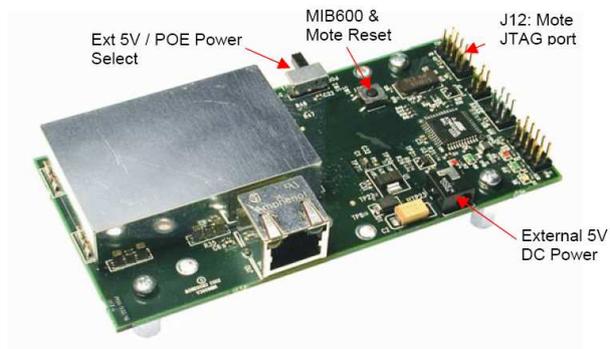


Figure A.5: MIB600 Ethernet Interface Module

### A.3 Stargate Processing module

The Stargate is a self-contained data processing platform that can directly interface with a network of Smart Dust motes, post-process the incoming data, and then transfer it to any external storage device through the Ethernet 10 Base-T LAN port, the Universal Serial Bus (USB) port, Bluetooth or via the mobile phone network using a Global System for Mobile Communications (GSM) module.

- Stargate System Software
  - Linux kernel 2.4.19
  - Open source distribution
  - Broad range of Familiar and Debian utilities
  - Support for jffs2, ext2, ext3, vfat, and msdos file types
  - 802.11 and Bluetooth wireless tools for NetGear, Xcomax, and Bluez
  - Mesh networking using AODV
  - POSIX with Java and Perl runtime environments

The device is based on the Intel XScale processor that is also used on a number of hand-held computers and Personal Digital Assistant (PDA) devices. The standard Stargate has a main processing module card and a daughter card that contains the communications ports and devices.

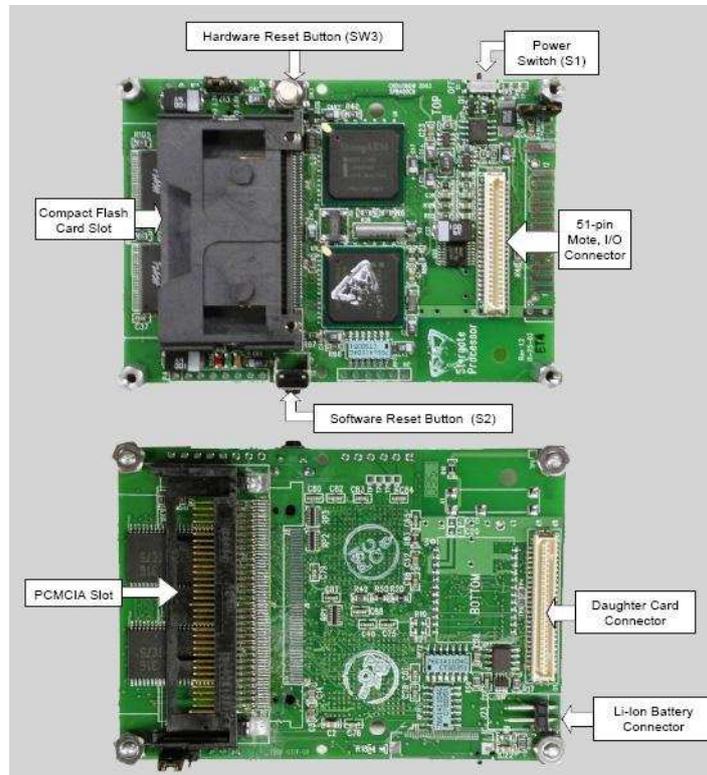


Figure A.6: Crossbow Stargate

- Stargate System Hardware
  - 400MHz, PXA55 XScale processor
  - 64 MB SDRAM, 32 MB Flash memory
  - 3.5 x 2.5 inches
  - MICA2(z) mote, Ethernet, Serial, JTAG, USB, PCMCIA, Compact Flash connectors
  - Bluetooth, 802.11 (through PCMCIA and Compact Flash Card)

A. CROSSBOW TECHNOLOGY™ SMART DUST SYSTEM

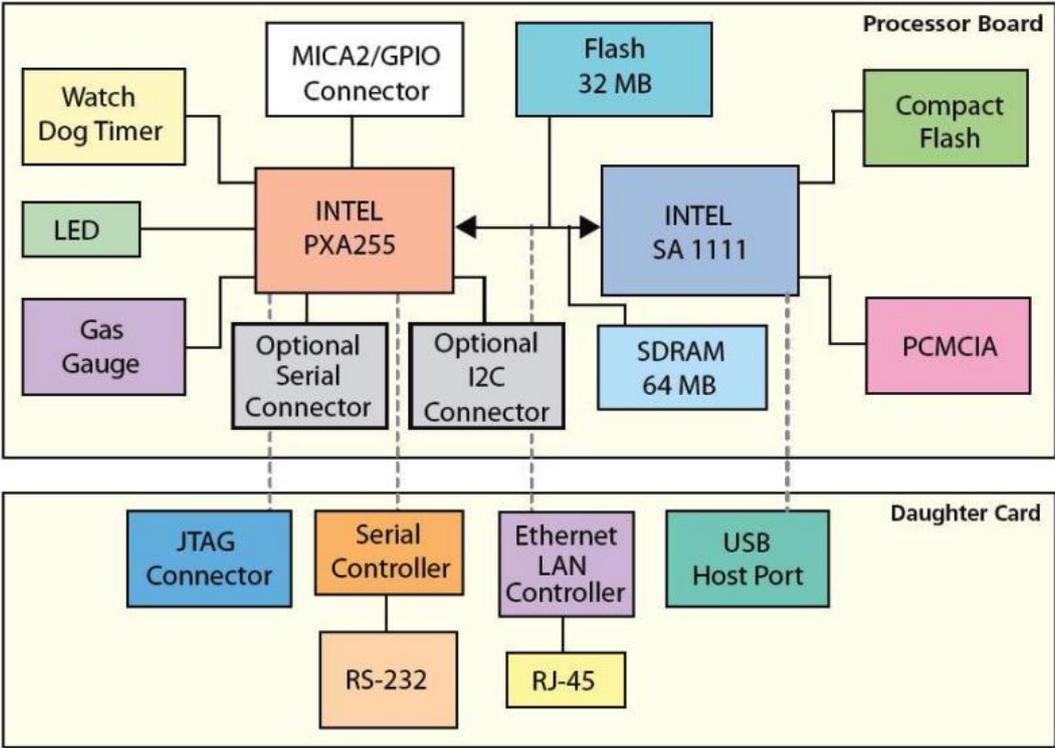


Figure A.7: Stargate functional diagram

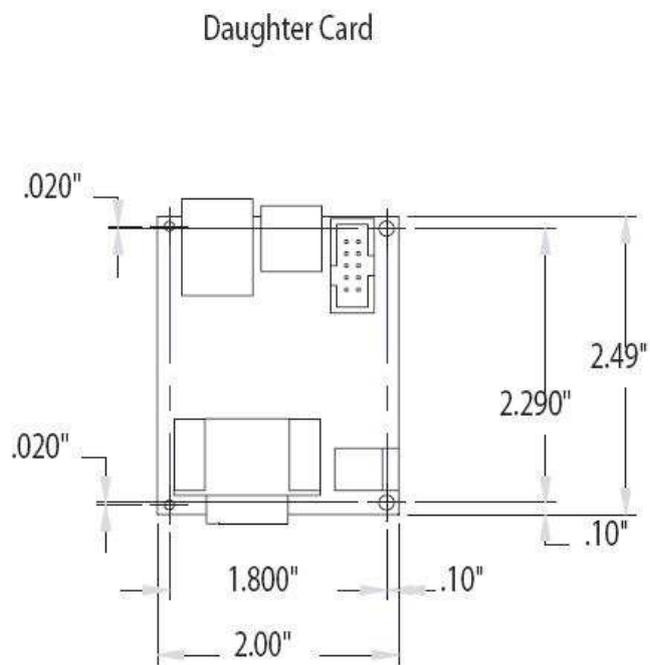
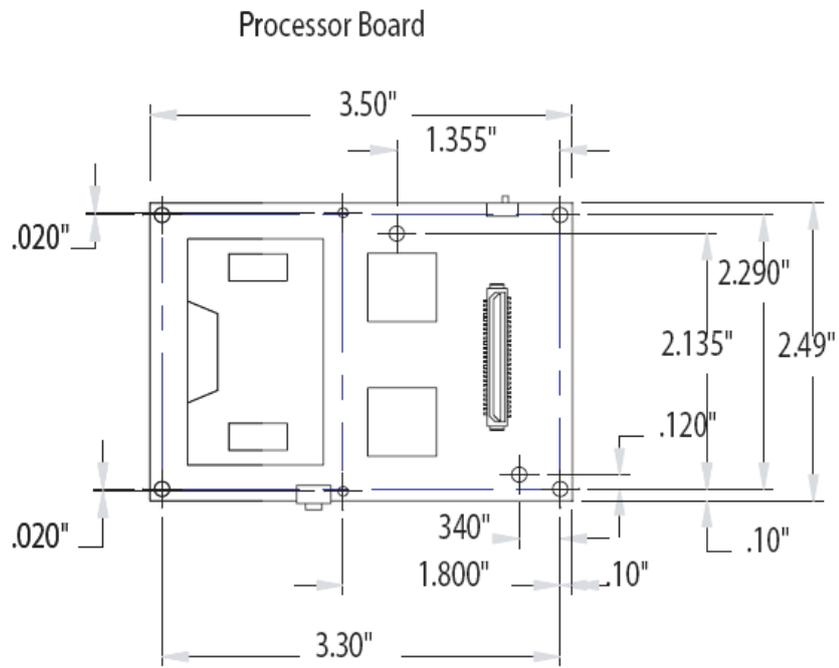


Figure A.8: Stargate dimensions

## A. CROSSBOW TECHNOLOGY™ SMART DUST SYSTEM

---

## Appendix B

# Smart Dust as an automatic weather station

This appendix illustrates, in a simple manner, how the various elements of a Smart Dust system can be configured to function as a synoptic automatic weather station, or SAWS in the terminology of the UK Meteorological Office. These generally take the form of a static installation using a set of wired sensors and fixed communications via a dedicated telephone line that directly links to a local Met. Office or through the telephone system to a more distant Met. Office. On some variants of the SAWS type systems there is the facility to manually add to, or modify, the automatic sensor readings.

### B.1 A simple automatic monitoring element

This is to illustrate the relative ease by which the Smart Dust element or mote can be configured to operate, electronically, as a SAWS device or even a standard Radio Sonde. The packaging, deployment hardware and a battery power supply capable of operating at very low temperatures are not considered here. By the addition of some off-the-shelf sensor components and the application of some standard operating system (TinyOS) and programming language (NesC) ADC code modules the mote can be made to operate as a generic SAWS device, whether it be a Radio Sonde type or a static type. In the case of the latter the mote can also be combined with an interface board, such as an MIB600 or XScale Stargate, which will handle the communications instead of the wireless system if required.

## B. SMART DUST AS AN AUTOMATIC WEATHER STATION

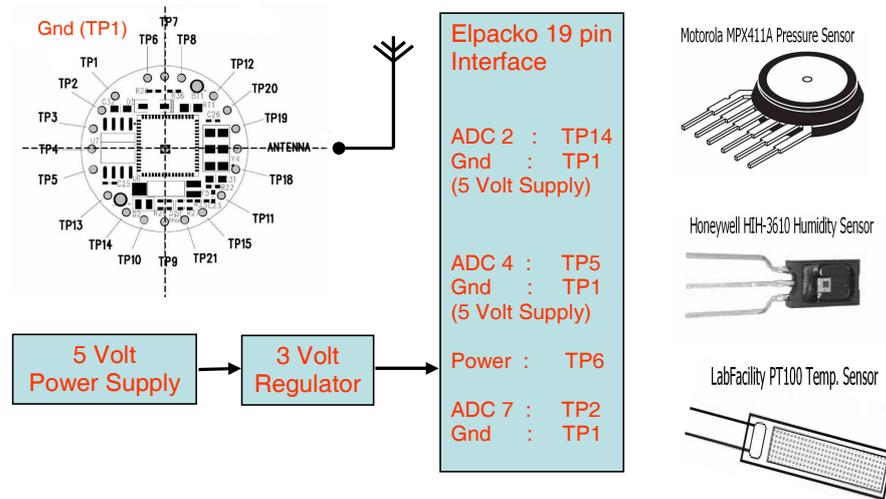


Figure B.1: Simple Smart Dust based Radio Sonde block diagram

- Motorola MPX4115A Pressure Sensor
  - Barometric pressure range 150—1150 hPa
  - 1.5% Maximum Error over 0°C—85°C
- Honeywell HIH-3610 Humidity Sensor
  - Relative Humidity range 0—100%
  - Relative Humidity accuracy  $\pm 2\%$
  - RH Linearity  $\pm 0.5\%$
  - Operating Temperature Range  $-40^{\circ}\text{C}$  —  $85^{\circ}\text{C}$  ( $-40^{\circ}\text{F}$  to  $185^{\circ}\text{F}$ )
  - 1.5% Maximum Error over 0 —  $85^{\circ}\text{C}$
- LabFacility PT100 Temperature Sensor
  - Platinum resistance element
  - Resistance ( $R_0$ ) 100 Ohms.
  - Class A temperature range  $-50^{\circ}\text{C}$  —  $+500^{\circ}\text{C}$

Figure B.2: Simple Smart Dust based Radio Sonde block diagram

The devices listed in figure B.2 have been selected to illustrate the general constitution of a stand-alone monitoring element. These devices are not intended as definitive as more appropriate, military specification devices, developed specifically for the task may be available - but probably at considerably higher cost. However, as shown in the

---

pressure calculation example in figure B.3 the pressure sensor and Smart Dust Analogue to Digital Converter (ADC) combination easily matches, and, outperforms the current commercially available Sonde devices used by the UK Meteorological Office.

### **B.1.1 Pressure Sensor and Analogue to Digital Converter example**

The lowest atmospheric pressure ever measured was 850 hPa, when adjusted to sea level set on 24<sup>th</sup> June 2003. The highest barometric pressure ever recorded was 1085.7 measured on 19<sup>th</sup> December 2001, this level of pressure is extremely rare however. The average barometric pressure for the planet is 1013.25 hectoPascals (Hpa), the MPX4115A pressure sensor can be configured to operate in a desired range, in this instance 800 to 1100 hPa more than covers the minimum and maximum pressure likely to be experienced on the planet.

#### ***Therefore:***

The MPX4115A range 800 – 1100 hPa (300 hPa)

The ATMega128L smart dust onboard 10 bit ADC has a range  $2^{10}$  which gives 1024 discrete values

$300/1024 = 0.292$  hPa (resolution per bit)

#### ***In Comparison:***

This very basic system provides a better resolution per ADC bit than these commercial Radio Sondes:

Vaisala RS80™ at 0.5 hPa

Vaisala RS90™ at 0.4 hPa

Figure B.3: Pressure Sensor and Analogue to Digital Converter example

### **B.1.2 A smart dust system sonde implementation**

Additional data types (instantaneous wind speed and direction vectors) can be inferred from the track that the Sonde takes during the flight. The tracking data necessary for this is generated by radar tracking or by the use of Global Positioning System modules on certain types of Radio Sonde. The single synoptic data stream produced can be considered as a limitation that could be improved upon by using a number of Radio Sonde devices in a cluster with the resultant multiple data streams combined in post-processing. This does pose some challenges in tracking multiple Radio Sondes simultaneously and ensuring that the wireless channels are discrete.

## B. SMART DUST AS AN AUTOMATIC WEATHER STATION

---

```
POWER UP
  AUTOMATIC SYSTEM INITIALISE
    Hardware boot up
    Operating system
    Duplex wireless system
    Mesh networking
  END SYSTEM INITIALISE
  RUN-TIME INITIALISE
    START: Internal storage database
    SET: Analogue to Digital Converters (2, 4 and 7)
    READ: Gateway setup data
      SET: Sample sensor timer (default: 2Hz)
      SET: Data transmission timer (default: 2Hz)
      SET: StartProcessing TO FALSE
    END READ:
  END RUN-TIME INITIALISE
  INFINITE LOOP                                (Standby stage)
    READ: StartProcessing from Gateway
    IF ( StartProcessing EQUALS TRUE )
      BREAK: INFINITE LOOP
    ENDIF
  END INFINITE LOOP                            (Standby stage)
  INFINITE LOOP                                (Run-time processing stage)
    IF ( Sample sensor Timer event )
      SAMPLE: ADC 2
      STORE: Pressure data
      SAMPLE: ADC 4
      STORE: Temperature data
      SAMPLE: ADC 7
      STORE: Humidity data
      BUILD: Data Synoptic data frame
      STORE: Data Synoptic data frame
    ENDIF
    IF ( Transmission timer event )
      IF ( Synoptic data frame(s) valid )
        BUILD: Data packet
      ELSE
        BUILD: Data packet using error data frame
      END-IF
    IF ( Packet Valid )
      SEND: Data packet to mesh networking sub-system
    ELSE
      SEND: Error packet to mesh networking sub-system
    ENDIF
  ENDIF
  END INFINITE LOOP                            (Transmission Timer event)
  END INFINITE LOOP                            (Run-time processing stage)
POWER END
```

---

The alternative approach to this development is the use of a cluster of wireless linked discrete Smart Dust sensor elements to monitor the 3-dimensional air-space. In this scenario each sensor element is configured as a discrete Radio Sonde which generate individual data streams each representative of a Radio Sonde device; the step-change to the data occurs when the discrete data streams are merged to form a 3-dimensional image of the monitored parameters from within the air-space. As previously stated the data produced has to be validated against a 3-dimensional geometric context.

A basic pseudo-code describing the run-time process of a standard Radio Sonde as implemented using the smart dust system is shown in figure [B.4](#).

### **B.1.2.1 An augmented Smart Dust based Radio Sonde**

A single Smart Dust device can be configured to function in a manner similar to a notional Radio Sonde, the requirement for this is the addition of a suite of appropriate sensors and bespoke software program to interrogate the sensor suite and transmit the resultant data stream. By the addition of a second Smart Dust device, configured as a receiver and logging module, a demonstrable, functional, single element Radio Sonde system can be created.

The next step is to build a cluster of Smart Dust devices, each configured as a Radio Sonde discrete sensor element as defined above. As a result of the diminutive form factor it is quite feasible to package them in an enclosure appropriate for the environment and subsequently deploy them as a defined cluster. In this mode they can be configured to monitor environmental parameters local to their largely unique physical location. The result is an enhanced Radio Sonde that, as a group, has the potential to generate multiple synoptic data streams. One major source of data corruption that must be avoided if at all possible is that of wireless transmission clashes, one way this limitation may be mitigated is by all devices acting as a group in order to synchronise wireless communications. In this example the receiver and logging module will now have the augmented task of collating these streams into a coherent data-sets of both the synoptic data and the associated RSSI data. With each cluster element monitoring its own 'local track' it now becomes possible, in theory, to monitor an environment in 3-dimensions, specifically, from the area covered by the physical dispersal of the Smart Dust devices. In order to facilitate this enhanced mode of operation the fundamental requirement becomes how to determine the physical position of each discrete sensor element at any given moment in time. One possible solution would be the addition of Global Positioning System (GPS) modules to the discrete sensor elements; whilst this method is used successfully on certain types of Radio Sonde it has a number of practical limitations. This approach will be examined, with specific emphasis on the practical and cost drawbacks that negate its use in a system using the aforementioned 'Smart Dust' based system. The elimination of GPS requires another method of establishing the physical location of the discrete Radio Sonde sensor elements.

## B. SMART DUST AS AN AUTOMATIC WEATHER STATION

---

### B.1.2.2 Clustered drop sonde pseudo-code

Figure B.5 is a very simple pseudo-code representation of the software required to configure a Smart Dust element into a clustered drop sonde.

#### POWER UP

##### AUTOMATIC SYSTEM INITIALISE

- Hardware boot up
- Operating system
- Duplex wireless system
- Mesh networking

##### END SYSTEM INITIALISE

##### RUN-TIME INITIALISE

- START:** Internal storage database
- SET:** Analogue to Digital Converters (2, 4 and 7)
- BROADCAST:** Unique mote hardware identifier
- READ:** Gateway setup data
  - SET:** Sample sensor timer (default: 2Hz)
  - SET:** Data transmission timer (default: 2Hz)
  - SET:** RSSI localisation timer (default: 1000Hz)
  - SET:** Cluster size
  - SET:** LocalRoundRobinIdent (in range 1 – Cluster size)
  - SET:** LastRoundRobinIdent to 0 (Static gateway)
  - SET:** StartProcessing TO FALSE

##### END READ:

##### END RUN-TIME INITIALISE

##### INFINITE LOOP *(Standby stage)*

- READ:** StartProcessing from Gateway
- IF ( StartProcessing EQUALS TRUE )**
  - BREAK: INFINITE LOOP**
- ENDIF**

##### END INFINITE LOOP *(Standby stage)*

##### INFINITE LOOP *(Run-time processing stage)*

- IF ( Sample sensor Timer event )**
  - SAMPLE:** ADC 2
  - STORE:** Pressure data
  - SAMPLE:** ADC 4
  - STORE:** Temperature data
  - SAMPLE:** ADC 7
  - STORE:** Humidity data
  - BUILD:** Data Synoptic data frame
  - STORE:** Data Synoptic data frame
  - IF ( Synoptic data frame invalid )**

---

```

        SET: Synoptic data frame to error data frame
    END-IF
ENDIF
IF ( Transmission Timer event )
    IF ( Synoptic data frame(s) valid )
        BUILD: Data packet
    ELSE
        BUILD: Data packet using error data frame
    END-IF
IF ( RSSI localisation timer event )
    IF ( NextRoundRobinIdent EQUALS LocalRoundRobinIdent )
        /* Transmit packet mode */
        BUILD: RSSI data frame from previously sampled data
    ELSE
        /* Receive mode - read RSSI data */
        BUILD: RSSI data frame
        READ: RSSI memory location
        READ: NextRoundRobinIdent (Source)
        READ: LocalRoundRobinIdent (Destination)
        READ: Time epoch ident
        READ: RSSI localisation event counter
        STORE: RSSI data frame
    END BUILD:
ENDIF
BUILD: Data packet using RSSI data frame synoptic data frames
IF ( Packet Valid )
    SEND: Data packet to mesh networking sub-system
ELSE
    SEND: Error packet to mesh networking sub-system
ENDIF
IF ( Last round-robin ident number LESS-THAN Cluster size )
    INCREMENT: NextRoundRobinIdent
ELSE
    SET: NextRoundRobinIdent to 0 (Static gateway)
ENDIF
END-IF
END INFINITE LOOP
POWER END

```

Figure B.5: Clustered drop sonde pseudo-code

## **B. SMART DUST AS AN AUTOMATIC WEATHER STATION**

## Appendix C

# The software model initialisation file

There is no definitive version of the initialisation file for the model due mostly to the developmental nature of the software, and also that the first section is simulation specific and could, conceivably, be empty. The version shown below represents one of the latest versions, as a result it contains the experimental parameters for all 'Single step' simulations that the model has been configured to run.

The file has two sections, simulation and generic system specific; the second section is effectively that which is defined the Virtual Base-class Framework specification. In practicality though it has become part of the Standard Derived Classes specification that is directly derived from the Virtual Base-class Framework.

### C.1 File section: Simulation specific parameters

This section of the current file contains the settings that are pertinent to the externally derived algorithm part of any simulation, this section is automatically read in and stored in a Virtual Base-class Framework data structure. The Virtual Base-class Framework specification requires a bespoke function implementation as part of the derived class structure to perform the parsing and any associated post-processing of this section of the input parameters.

The simulation specific section currently includes every parameter that has been used by any simulation type, as a result many of these parameters will be redundant for certain simulations; for example, the initial Random and Hill-climber simulations only require (and therefore parse) the parameter that controls the simulation type (`opAlgType`). This step was originally taken for the sake of simplicity, any future development of this software in principle or actuality would be to split the initialisation file into the two constituent parts as this would improve clarity and usability.

## C. THE SOFTWARE MODEL INITIALISATION FILE

---

### C.1.1 Generic simulation specific parameters

**Parameter :** expIdentFormat

Used to create simulation identifier that reflects the operators and other relevant parameters used, appended to the VBF generated output filenames. Each *Bit selector* that is specified (IDX\_NONE must be used on its own) will have its *Short ident* concatenated to a overall identifier string, in addition the status of what the *Bit selector* refers to will also be concatenated with the values (0—1) indicating false or true respectively.

Short ident	Bit selector	Description
NONE	IDX_NONE	No concatenation
NONE	IDX_DATA	Not used
NONE	IDX_DELIM	Not used
OpX	IDX_OPX	Crossover operator
OpM	IDX_OPM	Mutation operator
OpG	IDX_GW	Gateway active
RSw	IDX_RSW	Random swap
RJm	IDX_RJM	Random Mote Jump
GA	IDX_EA	Optimisation algorithm type
El	IDX_EL	Error level
ERR	IDX_ERR	Errors active

### C.1.2 Hybrid operator simulation parameters

The following group of parameters are all pertinent to a set of simulations that investigated a hybrid simulation operation mode. Some of the parameters here are no longer used and others are redundant due to becoming deprecated as they have been replaced or combined with other parameters that became more relevant in the context of the development of the simulation.

**Parameter:** exp\_OpHybridMode

This parameter controls the application of the hybrid operator mode.

Not active: Zero (false).  
Active: Any other value (true — positive or negative).

**Parameter:** exp\_OpHybridStartOperator

This parameter controls operator to use initially.

Not active: Zero (false).  
Constraint operator: Any other value (true — positive or negative).

---

**Parameter:** exp\_OpTogRndSelect

This parameter toggles the standard and constrained operators — not used.

**Parameter:** exp\_SwitchOnce

This parameter controls whether the operators are switched once only or can be switched back if required by algorithm. See also exp\_OpSwitchInterval.

Not Active                      Zero (false).  
Active:                            Any other value (true — positive or negative).

**Parameter:** exp\_OpHybridMaxSwitchVal

This parameter controls when the operator switch will take place. See also, exp\_OpHybridSwitchMoteFactor.

This must be in the range (0 – maxIteration).

**Parameter:** exp\_OpHybridSwitchMoteFactor

This parameter can be used to scale the switch point according to size of the current problem, ie the number of elements in the input data-set.

Use factor of 1.0:              Zero  
n \* element count:            Any other positive value (n).

**Parameter:** exp\_OpSwitchInterval

This parameter controls the number of iterations before next operator switch attempt after the initial switch — if exp\_SwitchOnce is false.

This must be in the range (0 – maxIteration).

**Parameter:** exp\_OpHybridSwitchRandomPC

This parameter allows for a random event to override the switch parameters listed above.

Not active:                      Zero  
Apply random switch:          Any value in the range (1 – 100%) iterations.

### C.1.3 Evolutionary algorithm parameters

This group are relevant to the general control of many parameters usually present in an optimisation algorithm that follows evolutionary principles.

**Parameter:**                      opTypeX

## C. THE SOFTWARE MODEL INITIALISATION FILE

---

This parameter controls The active crossover operator

Single Point	0
Random Uniform	1

**Parameter:** opTypeM

This parameter controls The active mutation operator

Mutation Swap	4 swap n pairs of XYZ mote positions
Shunt Section	5 shunt n motes section to new chromosome positions
Invert Section	6 reverse a section of n motes in chromosome
Move Mote all	7 Random XYZ positions for all chromosome motes
Move Mote N	8 Random XYZ positions for n chromosome motes
Move Mote Window	9 Random XYZ positions for n motes in a block

**Parameter:** exp\_OPXRandSelect

Crossover constraint random selection ratio)

Default	50/50%
Always select	100%
Never select	0%

This must be in the range (0–100%)

**Parameter:** allowRndSwap

This parameter controls the small random chance (10%) of a simple mote positional swap in addition to the active opTypeM operator.

Not active:	Zero (false).
Active:	Any other value (true - positive or negative).

**Parameter:** moteSwapMaxPercent

This parameter controls the number of motes to be selected for any

active swap type operator

Random number	Zero
Max number	0–50%

**Parameter:** allowRndMoteJump

This parameter controls the small random chance (10%) of a simple mote positional jump.

Not active:	Zero (false).
Active:	Any other value (true - positive or negative).

---

**Parameter:** moteJumpMaxPercent

This parameter controls number of notes selected for any active jump operator

Random number	Zero
Max number	0–50%

**Parameter:** moteJumpMaxDistPercent

This parameter controls the distance the random jump notes are allowed to jump. Random jump percentage 0–100% of XYZ bounds.

**Parameter:** mutateSelectPercent

This parameter controls the ratio be the application of the crossover operator and *accept unchanged* into the new chromosome population.

Default	70/30% This must be in the range (0–100%)
---------	---

**Parameter:** opAlgType

This parameter controls the active algorithm

Steady state	0 Single chromosome replacement EA
Part generational	1 Chromosome(n) replacement EA
Generational	2 Chromosome population replacement EA
Random (Progression)	3 Random algorithm - builds on previous iteration
Random (Reset)	4 Random algorithm — re-initialises every iteration
Hill-Climber	5 Simple stochastic hill-climber

**Parameter:** replaceMethod

This parameter applies to steady state and part generational algorithm types.

Tournament select	0
Replace random	1
Always replace worst	2
Random of 0, 1 or 2	4 Randomly select tournament/random/worst
Error)	3

**Parameter:** partialGenSelPercent

This parameter only applies to part generational algorithm type.

Single chromosome	1%
All chromosome	100% (this is the same as generational)
Error	0%
Percentage selected	1–100%.

## C. THE SOFTWARE MODEL INITIALISATION FILE

---

**Parameter:** `tournSelCountPercent`

This parameter controls the chromosomes count for tournament selection.

Out of range	0% Results in the default of 3 chromosomes
Percentage selected	1–25%.

**Parameter:** `elitismSelCountPercent`

This parameter controls the application of elitism to the selection and retention process when building a new population, this is only active for the generational algorithm.

No elitism	0%
Elite percentage	1–50%

**Parameter:** `exp_FitType`

This parameter controls the fundamental underlying algorithm that the EA simulations can apply, development dependent to an extent.

Std EA	1
Input error EA	2
Decomposition EA	3

### C.1.4 Evolutionary algorithm parameters: fixed origin constraint

These parameters are an addendum to the mutation operator selection parameters - if the fixed origin constraint is selected this automatically selects a different class hierarchy of operators that conform to the fixed origin constraint. The fixed point, at the XYZ Cartesian origin, is called the Gateway in the terminology of the set of simulations that this initialisation refers to.

**Parameter:** `exp_useGWConstraint`

This applies a fixed origin constraint to all positional mutations applied to all motes in all chromosomes.

Not active:	Zero (false).
Active:	Any other value (true — positive or negative).

**Parameter:** `exp_KeepStatsGW`

This parameter was originally used for debugging purposes, the output is a large set of files that track the changes to the mote XYZ Cartesian positions before and after each mutation event.

---

Not active: Zero (false).  
Active: Any other value (true — positive or negative).

## C.2 File section: Generic specific parameters

The generic section parameters have the most relevance as these pertain to the operation of the underlying Virtual Base-class Framework. Although some may be 'switched off' if not required for a particular simulation, a significant number are required and must contain valid information, such as the source and destination paths; these parameters are marked as compulsory.

### C.2.1 Debug parameters

**Parameter:** paranoiaLevelGatewayCheck

This parameter monitors the gateway for any alteration by any external code to its position and flags an error if it has been changed.

Not active: Zero (false).  
Active: Any other value (true — positive or negative).

**Parameter:** debugVerbose

This parameter controls the amount of on-screen text

Not active: Zero (false).  
Active: Any other value (true — positive or negative).

**Parameter:** pauseAtInitMessage

This parameter allows a review of the settings prior to a simulation run

Not active: Zero (false).  
Active: Any other value (true - positive or negative).

**Parameter:** stopOnInsignificantError

This parameter will stop the simulation on any level of initialisation error

Not active: Zero (false).  
Active: Any other value (true - positive or negative).

## C. THE SOFTWARE MODEL INITIALISATION FILE

---

### C.2.1.1 Generic System parameters: Mutation limitation control

**Parameter:** applyMutateLimit

This parameter applies perturbation modification to slow the movement of the elements when an optimal solution is being approached. When an element or zone is within applyElemPerturbPercent of the optimal fitness value the perturbation modification is triggered.

Not active: Zero (false).  
Active: Any other value (true — positive or negative).

**Parameter:** mutateLimitPercent

This parameter defines the trigger point of ideal fitness when to start limiting mutation, this is set by trial and error to match the problem under consideration.

No trigger 0%  
Trigger immediately 100%  
This must be in the range (0–100%)

**Parameter:** applyTheBrakesType

This parameter controls the point at which mutation limiting is applied - not used

Mote level 0  
Zone level 1

**Parameter:** triggerPointCount

This parameter controls the number of trigger or fitness step points to be applied when limitation is active.

A negative value will cause auto scale according to problem size.

**Parameter:** autoScaleTriggerTarget

This parameter is not used

**Parameter:** applyExponentToTrigger

This parameter applies an exponential function to the calculation of trigger step points, this is to further reduce the mutation limiting as the optimal fitness is approached, to effectively scale the limitation to the level of fitness.

Not active: Zero (false).  
Active: Any other value (true — positive or negative).

**Parameter:** showTrigReset

Additional debug parameter shows the trigger step point reset event on screen

---

Not active: Zero (false).  
Active: Any other value (true — positive or negative).

**Parameter:** showActiveLimitData  
Additional debug parameter shows the trigger step point values on screen

Not active: Zero (false).  
Active: Any other value (true — positive or negative).

### C.2.1.2 Generic System parameters: Input/output

Used to create simulation identifier that reflects the run-time and other relevant parameters used, appended to the Virtual Base Class generated output filenames. Each 'Bit selector' that is specified (ID\_NONE must be used on its own) will have its 'Short ident' concatenated to a overall identifier string, in addition the status of what the 'Bit selector' refers to will also be concatenated with the values (0—1) indicating false or true respectively.

Short ident	Bit selector	Description
NONE	ID_NONE	No concatenation
NONE—	ID_DATA	Not used
NONE	ID_DELIM	Not used
TEST	ID_TEST	Indicates test data
ML	ID_ML	Mutate limit
PC	ID_PC	Population count
It	ID_ITER	Maximum iterations
N	ID_N	Normalisation
PPc	ID_PPC	Perturbation percentage
SP	ID_SP	Starting position (Element initialisation)
Dt	ID_DATE	Date of run
Tm	ID_TIME	Time of run

**Parameter:** xB\_InputFileMode

## C. THE SOFTWARE MODEL INITIALISATION FILE

---

This parameter controls the dynamic or discrete input file mode, if set to dynamic then a set of files with contiguous numbering are expected. For discrete mode the file numbering is expected to be 0.

If discrete (standard) mode is selected each input data-set file is run for *n* runs and composite files produced, dynamic (or sequential) mode is where all files in a data-set are processed sequentially and then repeated for *n* runs.

Dynamic	Zero (false).
Discrete	Any other value (true - positive or negative).

**Parameter:** dryRun  
Debug parameter - controls the output file generation of the simulation

Dry run — no output	0
Perturb/no files	1
No perturb/write files	2
Perturb/write files	3 (Normal operation — default value)

**Parameter:** inputFileOrDir  
This parameter controls the type of input repository, if set to input file the contents of the file are read in and used as the names of the input data-set files, these are subject to any sub-directory data/file masking. If set to directory the contents of the directory are read, any sub-directory data/file masking data applied to generate file name.

Input File	Zero (false).
Directory	Any other value (true — positive or negative).

**Parameter:** expRootInputDir  
This parameter is used to define the root directory of the input data-set tree, sub-directories from this point are controlled by datasetSubDir.

A single dot (.)	present working directory.
#	replaced with expIdent

**Parameter:** datasetSubDir[1024]  
This parameter is used to define the sub-directory from the root directory of the input data-set tree.

**Parameter:** multiInFile  
This parameter controls the input file numbering and suffix control

Single input file	Zero (false) '*00000.sfx' used.
Multiple input file	Any other value (true — positive or negative).

---

**Parameter:** inFileTemplate

This parameter controls the input file numbering and suffix control, input file names built from ;template\_¿000000.sfx to ;template\_¿999999.sfx. The template can be any text that is valid in the context of the target operating system.

**Parameter:** fileTemplateStartNum

This parameter controls the first number to use for inFileTemplate\_nnnnnn, a negative number indicates use all files in the target directory starting from zero.

**Parameter:** fileTemplateEndNum

This parameter controls the last number to use for inFileTemplate\_nnnnnn, Ignored if fileTemplateStartNum is negative.

**Parameter:** expRootOutputDir

This parameter controls the Results output base/root directory name formatting, the string can be parsed to replace the following tokens.

'£'	replaced with "TEST"
'^'	replaced with expIdent.
'\$'	replaced with current date
'@'	replaced with current time
'!'	replaced with '\ ' or '/ '.

**Parameter:** outputSubDir

This parameter controls the Results output sub-directory from the base/root directory, the string can be parsed to replace the same tokens as defined for expRootOutputDir.

### C.2.1.3 Generic System parameters: Logging

**Parameter:** showRunLogToScreen

This parameter controls if run-time logs are printed to screen as well as the log file.

Not active:	Zero (false).
Active:	Any other value (true - positive or negative).

**Parameter:** showExpLogToScreen

This parameter controls if run-time logs are printed to screen as well as the log file.

Not active:	Zero (false).
Active:	Any other value (true - positive or negative).

**Parameter:** logFileInOutDir

## C. THE SOFTWARE MODEL INITIALISATION FILE

---

This parameter controls where the log files are placed.

In executable directory Zero (false).

Data output root dir Any other value (true — positive or negative).

### C.2.1.4 Generic System parameters: Program Run control

**Parameter:** fitDirection

This parameter controls the direction of optimal fitness

Maximising Zero (false).

Minimising Any other value (true — positive or negative).

**Parameter:** xyzInFile

Debug parameter - not used.

**Parameter:** processXYZFitness

Debug parameter - not used.

**Parameter:** popCount

This parameter controls the size of the population

**Parameter:** bestChromCount

Debug parameter - not used.

**Parameter:** allowSameMoteFitCalc

This parameter controls if mote to mote fitness calc with same ident allowed

Not active: Zero (false).

Active: Any other value (true — positive or negative).

**Parameter:** testDataStructs

Additional debug parameter

Not active: Zero (false).

Active: Any other value (true - positive or negative).

**Parameter:** testVal

Debug parameter - discrete value used as fitness value when testDataStructs true

**Parameter:** runtimes

This parameter control run count applied to identical data to generate average.

---

**Parameter:** maxIteration

This parameter controls the absolute number of iterations (not evaluations) performed per run, this may be overridden by any function useStopFitVal / stopFitVal that stops processing early if set.

**Parameter:** minIteration

This parameter controls the absolute minimum number of iterations (not evaluations) that will be performed per run, this cannot be overridden by any function useStopFitVal / stopFitVal that stops processing early if set.

Minimum iterations     1–99% of maxIteration

**Parameter:** useStopFitVal

This parameter enables the overriding of maxIteration

Not active:             Zero (false).

Active                     Any other value (true - positive or negative).

**Parameter:** stopFitVal

This parameter represents the trigger fitness value at which processing of the current run will stop if useStopFitVal is set.

**Parameter:** showAllFitVal

Debug parameter, controls whether all best fitness values are shown to screen.

Not active:             Zero (false).

Active:                     Any other value (true - positive or negative).

**Parameter:** dispFitProgress

This parameter controls how much of the run-time fitness processing is shown on the screen as a percentage of maxIterations.

Stepped display         1–99%

No display                0%

Display once at end     100%

**Parameter:** dispOnScreenMsg

This parameter controls whether run-time messages are shown on the screen

Not active:             Zero (false).

Active                     Any other value (true — positive or negative).

## C. THE SOFTWARE MODEL INITIALISATION FILE

---

### C.2.1.5 Generic System parameters: Normalisation

**Parameter:** `normalise_UnitCube`

This parameter controls the normalisation to Normalise the fitness data to a 1 unit cube to compensate for the XZY dimensions of the 3-dimensional airspace zone. If ID\_N is selected in expIdent then this shows up as N1 (or N3 if normalise\_ZoneCount is set).

Not active: Zero (false).  
Active: Any other value (true — positive or negative).

**Parameter:** `normalise_ZoneCount`

This parameter controls the normalisation of fitness data to compensate for the number of elements in the 3-dimensional airspace zone. If ID\_N is selected in expIdent then this shows up as N2 (or N3 if normalise\_ZoneCount is set).

Not active: Zero (false).  
Active: Any other value (true — positive or negative).

**Note. 1** If normalise\_UnitCube or normalise\_ZoneCount are true then a set of output files in sub-directory "Std\_PopFit\_Normalised" will be created in addition to the default non-normalised set of output files in another sub-directory "Std\_PopFit\_AsGenerated".

**Note. 2** Normalisation is not applied at runtime, all stored fitness data remains as generated until output to the results files.

### C.2.1.6 Generic System parameters: Mote start position settings

The following two lines control access to files containing XYZ triples from a previous run that can be used to initialise the positions of the motes. Each moteCount will have a file popv\_posData\_mc;n;.vtb where n = moteCount. They are normally stored in default directory {exp\_results\_dir}/expBPosData. If this directory does not exist or any particular file is missing then the run will stop readPosDataFromFile/writePosDataToFile are mutually exclusive - an error will occur if both are true

The parameter readPosDataFromFile overrides useStartPos and implies useSameStartPosForAll. The parameter writePosDataToFile writes the initialisation XYZ positions to the files that will become the files that a subsequent readPosDataFromFile will use. This parameter respects the useStartPos/useSameStartPosForAll settings.

**Parameter:** `readPosDataFromFile`

---

This parameter controls the reading of preset mote positions that have been stored in an external file that follows the naming convention given above.

Not active: NULL.  
popv\_posData\_mcjn<sub>i</sub>.vtb: Any other value (true — positive or negative).

**Parameter:** writePosDataToFile

This parameter controls the writing of initialised mote positions to an external file that follows the naming convention given above.

Not active: NULL.  
popv\_posData\_mcjn<sub>i</sub>.vtb: Any other value (true — positive or negative).

**Parameter:** externalPosDataDir

This parameter holds the directory of the external positional data file set in readPosDataToFile or writePosDataToFile.

**Parameter:** useSamePosOnRunReset

This parameter is used to control whether the motes are reset to the same XYZ position as set at the initialisation. if set true the mote init xyz positions are saved and used to re-initialise the population to the same settings at the beginning of a new run. New datasets always reset the initialisation settings.

Not active: Zero (false).  
Active: Any other value (true — positive or negative).

**Parameter:** useSamePosOnAllDataset

This parameter is similar to useSamePosOnRunReset except that the same settings as set at initialisation are used to re-initialise the population to the same settings to of a new run for ALL Datasets. This will only result in identical start pos fitness IFF the motecounts between datasets is the same, this parameter is ignored if useSamePosOnRunReset is set false.

Not active: Zero (false).  
Active: Any other value (true — positive or negative).

**Parameter:** useStartPos

This parameter controls how the motes initialisation positions are set (if they are not read in from the readPosDataFromFile). If this is set false then the XYZ positions are randomly set, otherwise the values in startPosX—Y—Z are used.

Ignore startPos\*: Zero (false).  
Use startPos\*: Any other value (true — positive or negative).

## C. THE SOFTWARE MODEL INITIALISATION FILE

---

**Parameter:** useSameStartPosForAll

This parameter controls how the individual notes are initialised, this is active IFF useStartPos is active (ie. Not overridden by readPosDataFromFile). If true then each note will have identical XYZ positions. If false then each will be generated individually.

Not active: Zero (false).

Active: Any other value (true — positive or negative).

**Parameter:** startPosX

The initialisation X axis value used if useStartPos is set true, must be in range — posMin..posMaxX.

**Parameter:** startPosY

The initialisation X axis value used if useStartPos is set true, must be in range — posMin..posMaxY.

**Parameter:** startPosZ

The initialisation X axis value used if useStartPos is set true, must be in range — posMin..posMaxZ.

**Parameter:** overrideInputFilePC

Not used.

**Parameter:** maxInitPercent

This parameter controls the initialisation perturbation value that each axis for each note is subject to after the initial XYZ position. Most use of all notes have identical start positions.

Max percent (1–99%) of max XYZ bounds.

**Parameter:** maxPerturbVariance

This parameter controls the maximum percent perturbation for all axes and for all notes in any mutation event.

Max percent (1–99%) of max XYZ bounds.

### C.2.1.7 Generic System parameters: fixed origin note start position settings

**Parameter:** allowGatewayMovement

This parameter controls whether the gateway note can move from the origin of the XYZ axes, currently this is an error if it does.

---

Not active: Zero (false).  
Active: Any other value (true — positive or negative).

**Parameter:** useDefGatewayPos

This parameter controls the position of the origin where the XYZ axes all cross, currently it is required that they all cross at the same point, the default value is 0,0,0.

Not active: Zero (false).  
Active: Any other value (true — positive or negative).

**Parameter:** gatewayX

This parameter controls the position of the origin X axis.

This must be between 0.0 and max X bound.

**Parameter:** gatewayY

This parameter controls the position of the origin Y axis.

This must be between 0.0 and max Y bound.

**Parameter:** gatewayZ

This parameter controls the position of the origin Z axis.

This must be between 0.0 and max Z bound.

### C.2.1.8 Generic System parameters: Results types

**Parameter:** StdOutFiles.Txt

This parameter controls if standard output text files in are generated

Not active: Zero (false).  
Active: Any other value (true - positive or negative).

**Parameter:** stdOutFiles.Bin

This parameter controls if standard output bin files are generated

Not active: Zero (false).  
Active: Any other value (true - positive or negative).

## C. THE SOFTWARE MODEL INITIALISATION FILE

---

### C.2.1.9 Generic System parameters: post processing to multi-run datasets

**Parameter:** mergeMultiRun

This parameter controls if multiple run data for the same data-set are built into tabulated files.

Not active: Zero (false).

Active: Any other value (true - positive or negative).

**Parameter:** keepMultiRunFiles

This parameter controls if multiple run files for the same data-set for each discrete run are kept, ignored if mergeMultiRun set false.

Not active: Zero (false).

Active: Any other value (true - positive or negative).

**Parameter:** bestXYZDataFile

Debug parameter - Not used

**Parameter:** avgPopFitResults

This parameter controls if average population fitness against iteration files are generated.

Not active: Zero (false).

Active: Any other value (true - positive or negative).

**Parameter:** avgchromFitResults

This parameter controls if average chromosome fitness against Iteration files are generated.

Not active: Zero (false).

Active: Any other value (true - positive or negative).

**Parameter:** avgMoteFitResults

This parameter controls if average mote fitness files against Iteration are generated.

Not active: Zero (false).

Active: Any other value (true - positive or negative).

### C.2.1.10 Generic System parameters: Multi-run dataset output options

**Parameter:** writeIntermediateCompoundFile

---

A debug parameter that controls if intermediate fitness files are written during the generation of multiple run data, useful if the run terminates for any reason as the last known state is preserved in the data.

Not active: Zero (false).  
Active: Any other value (true - positive or negative).

**Parameter:** writeDiscreteRunFiles  
This parameter controls if discrete run file are generated, as well as any merged data files.

Not active: Zero (false).  
Active: Any other value (true - positive or negative).

**Parameter:** writeBestFitVal\_AllRuns  
This parameter controls if best (normalised) fitness data files written.

Not active: Zero (false).  
Active: Any other value (true - positive or negative).

**Parameter:** writeRawFitVal\_AllRuns - not used  
This parameter controls if raw (non-normalised) fitness data files written.

Not active: Zero (false).  
Active: Any other value (true - positive or negative).

**Parameter:** writeElapsedTime\_AllRuns  
This parameter controls whether the elapsed time data files are written.

Not active: Zero (false).  
Active: Any other value (true - positive or negative).

#### C.2.1.11 Generic System parameters: Statistics types

**Parameter:** fileStatTypes  
This parameter controls the statistics columns that are contained in multi-run tabulated results files, the stats are calculated from the individual run results data columns.

Bit selector	Description
FMT_NONE	No stats
FMT_AVG	Average

## C. THE SOFTWARE MODEL INITIALISATION FILE

---

FMT_GRD	Gradient
FMT_MAX	Maximum
FMT_MED	Median
FMT_MIN	Minimum
FMT_SDV	Standard deviation
FMT_SUM	Sum
FMT_VAR	Variance
FMT_RGP	Range group

### C.2.1.12 Generic System parameters: Result file format types

**Parameter:** fileFormat\_CombinedExp  
This parameter controls the actual format of the results files.

Bit selector	Description
FMT_TAB	Use tabulatedstyle - override default single col
FMT_TTL	Titles above columns
FMT_DVEC_DATA	Include DVec column data in output files
FMT_VSTATS	Stats placed at bottom of column
FMT_HSTATS	Stats placed on the Right hand side of all columns
FMT_IDX	Add index to data
FMT_FSTATID	Append stats ident to filename
FMT_GNUPLOT	Add two lines between blocks of DVec/Stats (vertical) data as delimiter
FMT_STDFHDR	Keep raw data used for stats
FMT_KEEPPDAT	Write standard header to file

### C.2.2 Example Initialisation file

```
// ----- Simulation specific settings - Experiment control -----
// char *expIdentFormat - final value made up of above - space/tab
// delimited Ident Name types Or'd together - ID_NONE used alone
//IDX_NONE IDX_DATA IDX_OPX IDX_OPM IDX_GW IDX_RSW
//IDX_RJM IDX_GA IDX_EL IDX_ERR IDX_DELIM IDX_DATA IDX_OPX
//IDX_OPM IDX_GW IDX_GA IDX_EL IDX_ERR IDX_DELIM
// char *expIdentFormat - final value from above - space/tab delimited
0 // bool exp_OpHybridMode - 0 = false, non-zero = true.
0 // bool exp_OpHybridStartOperator - 0 = Standard, non-zero = Constraint type.
// bool exp_OpTogRndSelect - 0 = auto-toggle operators, non-zero = rand select
// Line below overrides random switch events
// (ignores exp_OpHybridSwitchMoteFactor/exp_OpSwitchInterval/
// exp_OpHybridSwitchRandomPC and exp_OpTogRndSelect)
// Switches exp_OpHybridStartOperator at exp_OpHybridMaxSwitchVal
0 // bool exp_SwitchOnce - 0 = false, non-zero = true.
1000 // long exp_OpHybridMaxSwitchVal
```

---

```

// 2 Lines below control the point at which the operator switch is performed
// switchScaleFactorMoteCount * number of motes
// scales according to problem complexity - default = 100.
100 // long exp_OpHybridSwitchMoteFactor - 0 = use factor of 1.0, else n * mote
100 // long exp_OpSwitchInterval - iterations before next switch attempt after initial
// switch
0 // long exp_OpHybridSwitchRandomPC (0..100%) - zero = always switch selected
// pos val = random switch
0 // long opTypeX: Crossover Single Point(0), Crossover Random Uniform(1)
7 // long opTypeM: Mutation Swap(4);, Shunt Section(5), Invert Section(6),
// Move Mote Std(7), Move Mote Rnd(8), Move Mote Rnd WIndow(9)
0 // bool useGWConstraint: 0 = false, non-zero = true. Use Movemote Gateway
// Constraint operators
0 // bool exp_KeepStatsGW: 0 = false, non-zero = true (and useGWConstraint = true).
// Record GW stats - debug mode
30 // long exp_OPXRandSelect Crossover constraint random selection ratio (0..100%)
//default = 50/50% - 0 = Always select 100% = never select
1 // bool allowRndSwap - 0 = false, non-zero = true. small chance (10%) of simple swap
// in addition to opType = Mutation Swap,
30 // long moteSwapMaxPercent - zero = random number selected (0..50%). Max
//number of motes to swap
1 // bool allowRndMoteJump - 0 = false, non-zero = true. small chance (10%) of mote
// jump if true
30 // long moteJumpMaxPercent - zero = random number selected (0..50%). Max number
// of motes allowed to jump
99 // long moteJumpMaxDistPercent - zero = random jump percentage selected (0..100%).
// Max distance motes (above) allowed to jump
40 // long mutateSelectPercent - Crossover/Random unchanged selection ratio (0..100%)
// default = 70/30%
// Line below - if set to 0 or 2 partialGenSelectPercent setting ignored
// if set to 1 the partialGenSelectPercent setting used to control the GA
0 // long opAlgType - steady state(0), part generational(1), generational(2),
// Random(3), Random_Reset(4), Hill-Climber(5)
// Line below - only applies to steady state/part generational 0 = tourn select,
// replace random(1), always replace worst(2), random of 0, 1 or 2 (4) error(3)
1 // long replaceMethod
25 // long partialGenSelPercent - zero = error (1..100) the percentage of chroms
// in pop selected. 1% = 1 chrom, 100% = all - same as generational opAlgType
5 // long tournSelCountPercent - zero = error, (1..25%) the percentage of chroms
// in pop selected. Zero causes default. 1% = 1 chrom
// Elitism is only active for the generational opAlgType
0 // long elitismSelCountPercent: zero = no elitism, (0..50%) the % of chroms selected.
1 // long exp_FitType - std(1), errors(2), Decomposition(3)

// ----- Decomposition Fitness object params -----
10 // long exp_DecompSizeSC - max Size of the Decomp sub-config (Max = 10)
1 // long exp_SubConfigOffset - default = 1 to offset for gateway - Range =
// (0..min motecount size (6))
20.0 // double exp_TargetFitVal_SC - target fitness val for single sub-config
// Line below - controls the reconstruction mode after the S-C stage.
0 // long exp_ReconMode - 0 = Use standard operators, 1 = use SC_Operators
// Line below - point at which fitness mode (S-C to Std) will switch regardless
// of S-C results negative value for no auto switch

```

## C. THE SOFTWARE MODEL INITIALISATION FILE

---

```
1500 // long autoSwitchIteration
    // Line below - Set the min percentage (total S-C for all chroms) at which the
    // fitness type switch will be forced. 100% for no auto switch - requires all S-C to
    // be within exp_TargetFitVal_SC fitness
80 // long exp_MinSwitch_PC_SC - Min percentage of S-C within fitness before switch

// ----- Error Fitness object params -----
    // The following lines are only active if exp_FitType = errors. They give the min
    // number of total packets =  $i$  0.0% = 1 packet min, total cannot be greater than 60%
0 // bool correctErrorOnceOnly - 0 = false, non-zero = true
0 // bool dontAllowGatewayError - 0 = false, non-zero = true
0 // bool allowMissingPktReverseReplacement - 0 = false, non-zero = true
60 // double pktMissing_pc: zero = percentage number selected (0..60% **).
    // Max number of missing packets
0 // double pktLowPayload_pc: zero = percentage number selected (0..60% **).
    // Max number of packets with reduced payload
0 // double pktHighPayload_pc: zero = percentage number selected (0..60% **).
    // Max number of packets with increased payload

// ----- Generic System parameters - Debug control -----
    // Line below only active if dispOnScreenMsg is true
0 // bool paranoiaLevelGatewayCheck -  $i$  0 = false, non-zero = true
0 // bool debugVerbose -  $i$  0 = false, non-zero = true
    // Line below only active if dispOnScreenMsg is true
0 // bool pauseAtInitMessage -  $i$  0 = false, non-zero = true
1 // stopOnInsignificantError -  $i$  0 false, non-zero = true

// ----- Generic System parameters - Mutation limitation control -----
    // Line below - apply perturbation modification to slow the movement of elements
    // when a fit solution is being approached. When an element/zone is within
    // applyElemPerturbPercent of best fitness - perturbation modification triggered
0 // bool applyMutateLimit -  $i$  0 = false, non-zero = true.
    // Line below - trigger point of ideal fitness when to start limiting mutation
    // Where 100% = the best fitness possible/No trigger (in effect)
    // 0% = No trigger
99.5 // double mutateLimitPercent
1 // long applyTheBrakesType = 0 = at Element level, 1 = at Zone level
    // Line below - number of trigger points - best  $\approx$  15
    // negative number = auto scale according to problem size - ie. larger worst fitness
    // results in larger number of points. Too many points at low worst fitness values results
    // in many fractional trigger points that are unlikely to be used
    // (ie.  $2.29371e - 07$  mFact = 0.48)
-20 // long triggerPointCount: (1..20) number mutation limiting divisions after trigger point
    // Line below - Used to calculate number of trigger points when triggerPointCount is
    // negative directly controls autoScaleDivExp() function generates the appropriate
    // number of steps based on the problem size. Larger number gives fewer steps.
    // Ignored if triggerPointCount positive.
0.005 // double autoScaleTriggerTarget - used to control autoScaleDivExp
1 // bool applyExponentToTrigger -  $i$  0 = false, non-zero = true.
0 // bool showTrigReset -  $i$  0 = false, non-zero = true. - Debug
0 // bool showActiveLimitData -  $i$  0 = false, non-zero = true. - Debug

// ----- Generic System parameters - Input Data Sets -----
```

---

```

// Ident Name types Or'd together - ID_NONE must be used alone
// ID_NONE ID_DATA ID_ML ID_PC ID_ITER ID_N ID_PPC ID_SP ID_DATE
// ID_DELIM ID_DATA ID_ML ID_PC ID_PPC ID_DELIM // char *expIdentFormat
// char array expIdent - MAXRUNIDLEN = 256 - Experiment identifier,
// expIdentFormat appended if length of expIdent >0
exp_StdEA_450-500
// Following line - controls dryRun/output
// Legal values 0 = dry run - no output, just tests the program structure
// Perturb/write files(3) (default), normal operation(3) ,no perturb/write files(2),
// Perturb/no files(1), (>3)=ERROR
// Line below - true = Discrete (standard) mode - each file run for 'N' runs and
// composite files produced False - Dynamic, Sequential mode - files processed
// sequentially and repeated for 'N' runs
1 // bool xB.InputFileMode ->0 = false, non-zero = true.
3 // long dryRun 3,2,1
0 // bool inputFileOrDir -i 0 = false (inputFile), non-zero = true (directory)
// 0 = use inputRunIdentFile, 1 (non zero) = use inputBaseDataDir directly
// A single dot (.) = pwd. Use '!' for '\ ' or '/'. Full dos path ok (d:\etc)
// # replaced with expIdent
// Line below - char array iPATHiinputRunIdentFile - MAXRUNIDLEN = 2048
// Active if inputFileOrDir = 0
// The file contains the path/dir of the input dataset(s) root location(s)
// File generated - dir /AD /B >datlist.txt or ls -d */ >datlist*.txt
/work/PhD/Model/InputDataSets/RndInitPosDS_200500_ipc_0.10_ppc_10.00/datlist.txt
// Line below - char array expRootInputDir[1024] - input base dir.
// Active if inputFileOrDir = 1
// The dir contains the directories of the input dataset(s)
/work/PhD/Model/InputDataSets/RndInitPosDS_200500_ipc_0.10_ppc_10.00/
// Line below - char array datasetSubDir[1024] ! and # can be used
// If set this is appended to the currently active input data set directory
// Set to NULL (all uppercase) if not required time.files
0 // bool multiInFile: 0 = false (single file '*00000.sfx' only), non-zero = true (*nnnnn.sfx)
time_// char inFileTemplate[256] - input file names built from itemplate_i000000 to 999999.
0 // long fileTemplateStartNum - first number to use for inFileTemplate_nnnnnn. neg = all
3 // long fileTemplateEndNum - the last number to use for inFileTemplate_nnnnnn.
// Ignored if above is neg
0 // bool allInputFilesSameDef ->0 false, non-zero = true

// ----- Generic System parameters - Output Root directory -----
// '£' replaced with TEST replaced with expIdent. '$' replaced with Date
// '@' replaced with time '!' replaced with '\ ' or '/'.
// Single dot (.) = pwd. Full dos path ok (d:\etc)
// Line below - char expRootOutputDir[1024] - Results output base dir.
/work/PhD/Results_Src/EA_STD/LargeDataset/
// Line below - char outputSubDir[256] - base dir. A single dot (.) = pwd.
// Use '!' for '\ ' or '/'. # replaced with expIdent.
NULL
1 // bool useInDataSetForOutDir ->0 false, non-zero = true

// ----- Generic System parameters -----
// Line below expRunLog filename. Used for runtime/init messages
// Log file will be placed in the outputDataRootDir
# // # alone = use expIdent(_run.log), NULL (all uppercase) = use default logfile

```

## C. THE SOFTWARE MODEL INITIALISATION FILE

---

```
1 // long showRunLogToScreen ->0 = false, non-zero = true
// Line below expResLog filename. Used for runtime/init messages
// Log file will be placed in the outputDataRootDir
# // # alone = use expIdent(.exp.log), NULL (all uppercase) = use default logfile
1 // long showExpLogToScreen ->0 = false, non-zero = true
1 // bool logFileInOutDir ->0 = false, non-zero = true. 0=leave in exe dir,
// 1 = move to new dataset out dir
1 // bool fitDirection ->0 = MAXIMISING, non-zero = MINIMISING.
1 // bool xyzInFile ->0 = false, non-zero = true. False = input files have no xyz data
0 // bool processXYZFitness ->0 = false, non-zero = true. If true needs above true
50 // long popCount - number of discrete 'airspace' (chromosomes) in the population
10 // long bestChromCount - number of best chroms kept during processing: min = 1
// Line below - if true then fitness calc between motes with same ident allowed
0 // bool allowSameMoteFitCalc ->0 = false, non-zero = true

// -----Generic System parameters - Program Run control -----
0 // bool testDataStructs ->0 = false, non-zero = true.
10.0 // double testVal ->value used when above is true
10 // long runTimes - the number of times the program will process the same data
10000 // long maxIteration
// Line below minimum iterations always performed if useStopFitVal = true
// regardless of stopFitVal
// n = specific iterations, n% = 1-99% of maxIteration
1000 // long minIteration
0 // bool useStopFitVal ->0 false, non-zero = true
5.0 // double stopFitVal ->stops processing if fitness  $\leq$  stopFitVal
0 // bool showAllFitVal ->0 false, non-zero = true
10.0 // double dispFitProgress: Range: (0-100%), no display(0%), display at end(100%)
1 // bool dispOnScreenMsg ->0 = false, non-zero = true

// ----- - Input dataset params - Mote start position settings -----
// No/Both normalisation types ->N0/N3 unit cube -i N1 : element zoncount -i N2
1 // bool normalise_UnitCube - Normalise the fitness data to a 1 unit cube -i N1
1 // bool normalise_ZoneCount - Normalise the fitness data to compensate for
// the number of elements in zone ->N2
// If normalisationActive is true output files in dir "Std_PopFit_Normalised"
// created in addition to the default non-normalised set of output files in dir
// "Std_PopFit_AsGenerated"
// Normalisation not used at runtime, stored data remains unchanged
7 // long activeAxes - Legal values 7=XYZ, 6=YZ,5=XZ,3=XY,
// 4=Z,2=Y,1=X, 0=NONE=ERROR
// The following two lines control access to files containing XYZ triples from a
// previous run that can be used to initialise the positions of the motes.
// Each moteCount will have a file popv_posData.mc;n.vtb where n=moteCount they
// are normally stored in default directory  $\text{exp\_results\_dir}/\text{expBPosData}$ .
// If directory does not exist or any particular file is missing then run will stop.
// readPosDataFromFile/writePosDataToFile mutually exclusive - both true = error.
// readPosDataFromFile overrides useStartPos and implies useSameStartPosForAll
// writePosDataToFile writes the first set of XYZ positions - DOES USE
// useStartPos/useSameStartPosForAll
0 // bool readPosDataFromFile ->0 = false, non-zero = true.
0 // bool writePosDataToFile ->0 = false, non-zero = true.
// Use NULL to signify read/write data to expRootOutputDir/expBPosData
```

---

```

// Line below - char externalPosDataDir[1024] - directory of external positional data file
/Work/PhD/Model/Results/TEST_OP001.2.2//
// Line below - if set true mote init xyz positions saved and used to re-initialise the
// population
// to the same settings at the beginning of a new run on the SAME data set.
// New datasets always start fresh
1 // bool useSamePosOnRunReset ->0 = false, non-zero = true.
// True = pop init mote pos data saved
// Line below - if set true mote init xyz positions saved and used to re-initialise the
// population to the same settings at the beginning of a new run on ALL Datasets
// Will result in identical start pos fitness IFF the motecounts between datasets same
// Ignored if useSamePosOnRunReset == 0
1 // bool useSamePosOnAllDataset ->0 = false, non-zero = true.
// True = pop init mote pos data saved
1 // bool useStartPos ->0 = false, non-zero = true. False ignores startPos*
// values below Line below will ensure that the initial xyz positions are the same for
// all motes - these values will be random if useStartPos is false or will be set to
// startPos[XYZ] if useStartPos is true
1 // bool useSameStartPosForAll ->0 = false, non-zero = true. Ignored if useStartPos false
// initial start XYZ pos for all motes - before maxInitPercent applied
500.0// double startPosX - posMin..posMaxX. Random pos use j posMin.
1000.0// double startPosY - posMin..posMaxY. Random pos use j posMin.
500.0// double startPosZ - posMin..posMaxZ. Random pos use j posMin.
0 // bool overrideInputFilePC ->0 = false, non-zero = true. False ignores max*Percent*
// values, uses input file hdr data
99 // double maxInitPercent - percent (1-99%) max init mote spacing - to max XYZ bounds
7 // double maxPerturbVariance - percent (1-99%) movement in any one time slot - to max
// XYZ bounds

// ----- - Input dataset params - Gateway Mote start position settings -----
0 // bool allowGatewayMovement ->0 = false, non-zero = true.
1 // bool useDefGatewayPos ->0 = false, non-zero = true. False ignores gateway* values
0.0 // double gatewayX
0.0 // double gatewayY
0.0 // double gatewayZ

// ----- - Generic System parameters - Results types -----
// if both false then dry run results
1 // bool StdOutFiles.Txt ->0 = false, non-zero = true.
1 // bool stdOutFiles.Bin ->0 = false, non-zero = true.

// ----- - Additional post processing to multiple run datasets -----
1 // bool mergeMultiRun - Merge Run data from same time slots
0 // bool keepMultiRunFiles - Keep/discard discrete run files - ignored if above set false
0 // bool bestXYZDataFile - Output XYZ params for best bestChromCount chroms
1 // bool avgPopFitResults - Average population fitness against Iteration
0 // bool avgChromFitResults - Average Chrom fitness against Iteration
0 // bool avgMoteFitResults - Average Mote fitness against Iteration

// ----- - Multiple run datasets - output options -----
1 // bool writeIntermediateCompoundFile
0 // bool writeDiscreteRunFiles
1 // bool writeBestFitVal.AllRuns

```

## C. THE SOFTWARE MODEL INITIALISATION FILE

---

```
0 // bool writeRawFitVal_AllRuns
1 // bool writeElapsedTime_AllRuns

// ----- Stats types -----
// FMT_AVG, FMT_GRD, FMT_MAX, FMT_MED, FMT_MIN, FMT_SDV,
// FMT_SUM, FMT_VAR FMT_SDIFF FMT_RDIF FMT_SIDX FMT_RGP()
// char *fileStatTypes - final value made up of above - space/tab delimited
FMT_AVG
FMT_MAX
FMT_MIN
FMT_SDV
FMT_RGP(0;j=2.5;2.5;j=5;5;j=10;10;j=50;50;j=1000;)
// ----- Format types -----
// Following controls the format of the Best/Raw output files for each experiment
// FMT_TAB, FMT_TTL, FMT_DVEC_DATA, FMT_VSTATS, FMT_HSTATS
// FMT_IDX, FMT_FSTATID, FMT_GNUPLOT, FMT_STDFHDR,
// FMT_KEEPPDAT, FMT_VSTATS, FMT_HSTATS stats as defined by fileStatTypes
// The minimum value is FMT_DVEC_DATA - this is optional if others are set
// char *fileFormat_CombinedExp - value made up of above - whitespace/tab delimited
FMT_TAB
FMT_TTL
FMT_HSTATS
FMT_FSTATID
FMT_STDFHDR
```

## Appendix D

# The software input data-set file

The format of the results files is bespoke to the software model used in this thesis, the basic format is shown in figures D.1 & D.2, the point behind the original design is that the data block reflects the sort of data stream that would be expected from a series of data-packets such as those found on any computer network. The header blocks are optional to support this concept should the files be replaced with a dynamically generated stream of packets (as in a network) or a Unix style pipe.

Field	Description
Type	Text packet type Identifier; "NULLPKT" "SYNCPKT" "RSSIPKT"
Ident	Numeric value of packet type Identifier; (See definitions below)
Source	Unique ident from SENDING element (the sender of this data packet)
Remote	Unique ident from REMOTE element (data packet receiver, adds RSSI/SYNC)
Total	Total number of packets for this (originating) element
Time	Globally set time epoch identifier
Payload	RSSI value in text form or GW_BROADCAST/RR_BROADCAST
End	End of packet marker - Currently set to ','

**Definitions:**

Identifier	Value	Description
NULLPKT	0L	Missing data
SYNCPKT	1L	Gateway packet to cause all element synchronisation
RSSIPKT	2L	Packet containing RSSI data
GW_BROADCAST	100L	RSSI data originates from the gateway
RR_BROADCAST	101L	RSSI data originates from a remote element

Figure D.1: Synchronised Inter-element Packet Format Description

## D. THE SOFTWARE INPUT DATA-SET FILE

---

Field identifiers								
Field	Type	Ident	Source	Remote	Total	Time	Payload	End
BYTES	12	4	4	4	4	4	30	4
TYPE	Byte array	Enumerated	Integer	Integer	Integer	Double	Byte array	Byte array

Figure D.2: Synchronised Inter-element Packet Format

### D.1 Header Blocks.

The header blocks are optional and give a range of information, most of which describe the parameters used to generate the data in the following data block (marked gen). Some parameters are used for debugging and checking purposes (marked chk), others can be used to control the subsequent model processing (marked ctl), although this has been deprecated in the model due to the optional nature of the header blocks.

#### Parameter Value Description Use

<b>Control Header ::</b>		Start of control header token	ctl
<b>main_Hdr_present</b>	1 (True)	Declares presence of header	ctl
<b>param_Hdr_present</b>	0 (False)	Declares presence of header	ctl
<b>absXYZ_present</b>	1320 (True)	Declares presence of data type	ctl
<b>relXYZ_present</b>	1320 (True)	Declares presence of data type	ctl
<b>hdrCount</b>	3	Numeric equivalent of above	chk
<b>Main Header ::</b>		Start of main header token	ctl
<b>runIdent</b>	DS_m010	Data-set identifier string	gen
<b>moteCount</b>	10	Number of motes	gen/ctl
<b>timeSecs</b>	10	Time covered by data-set	gen
<b>intPerSec</b>	1.00	Intervals per second	gen/ctl
<b>syncInterval</b>	1.00	Frequency of synchronisation	gen
<b>units</b>	METRE_0	Notional unit	ctl
<b>posMin</b>	0.00	Airspace zone XYZ origin	gen
<b>posMaxX</b>	1000.00	Airspace zone X bound	gen
<b>posMaxY</b>	1000.00	Airspace zone Y bound	gen
<b>posMaxZ</b>	1000.00	Airspace zone Z bound	gen/ctl
<b>startPosX</b>	304.71	Starting X position for motes	gen/ctl
<b>startPosY</b>	664.82	Starting Y position for motes	gen/ctl
<b>startPosZ</b>	15.27	Starting Z position for motes	gen/ctl
<b>useStartPos</b>	False	Not used	
<b>maxInitPercent</b>	0.10	Initial perturb X Y Z	gen/ctl
<b>maxPerturb</b>	10.00	Perturbation from startPosX Y Z	gen/ctl
<b>gatewayX</b>	0.00	Starting X position for Gateway	gen/ctl
<b>gatewayY</b>	0.00	Starting Y position for Gateway	gen/ctl
<b>gatewayZ</b>	0.00	Starting Z position for Gateway	gen/ctl
<b>useGatewayPos</b>	True	Use above for fixing Gateway	gen/ctl
<b>allowGatewayMovement</b>	False	Gateway XYZ not changed	gen/ctl
<b>activeAxes</b>	7	X=1, Y=2, Z=4. All axes=7	gen/ctl
<b>loadInputXYZ</b>	True	Not used	
<b>relXYZ_InDefFiles</b>	True	Not used	

---

<b>absXYZ_InDefFiles</b>	True	Not used	
<b>relXYZ_InTimeFiles</b>	True	Relative XYZ in data block	ctl
<b>absXYZ_InTimeFiles</b>	True	Absolute XYZ in data block	ctl
<b>relXYZ_InMoteFiles</b>	False	Not used	
<b>absXYZ_InMoteFiles</b>	False	Not used	
<b>outputDefFiles</b>	True	Output default fitness files	ctl
<b>outputXYZFiles</b>	True	Output XYZ results data files	ctl
<b>outputTimeFiles</b>	True	Output the elapsed time data files	ctl
<b>outputMoteFiles</b>	False	Not used	
<b>outputDeltaFiles</b>	False	Not used	
<b>resyncPktCount</b>	10	How many sync packets	chk
<b>resyncResponses</b>	100	How many sync responses	chk
<b>broadcastPktCount</b>	110	Gateway broadcast packet count	chk
<b>roundRobinRecCount</b>	1100	Gateway to mote packet count	chk
<b>nullPktCount</b>	0	Not used	
<b>totalPktCountInFile</b>	132	Total packets including Gateway	chk
<b>pktPerTimeSlot</b>	132	Not used	
<b>pktPerMote</b>	110	Packets sent per mote	chk

### D.1.1 Data Block.

pPktT:pktType:pktSource:pktSender:pktTotal:timeIdent:payload:endChar:  
(Optional-Abs(X, Y, Z, t) Rel(X, Y, Z, t))

Abs-0(0,0,0,0)	Rel-0(0,0,0,0)
Abs-1(780.255,705.593,924.523,0)	Rel-1(780.255,705.593,924.523,0)
Abs-2(933.405,509.883,201.967,0)	Rel-2(933.405,509.883,201.967,0)
Abs-3(159.744,429.058,140.724,0)	Rel-3(159.744,429.058,140.724,0)
Abs-4(5.2562,392.447,898.676,0)	Rel-4(5.2562,392.447,898.676,0)
Abs-5(610.037,749.891,965.386,0)	Rel-5(610.037,749.891,965.386,0)
Abs-6(972.232,654.206,424.028,0)	Rel-6(972.232,654.206,424.028,0)
Abs-7(484.483,250.781,769.826,0)	Rel-7(484.483,250.781,769.826,0)
Abs-8(126.599,35.5006,666.911,0)	Rel-8(126.599,35.5006,666.911,0)
Abs-9(657.245,353.739,598.14,0)	Rel-9(657.245,353.739,598.14,0)
Abs-10(304.523,664.499,14.8264,0)	Rel-10(304.523,664.499,14.8264,0)
Abs-0(0,0,0,0)	Rel-0(0,0,0,0)
Abs-1(780.255,705.593,924.523,0)	Rel-1(780.255,705.593,924.523,0)
Abs-2(933.405,509.883,201.967,0)	Rel-2(933.405,509.883,201.967,0)
Abs-3(159.744,429.058,140.724,0)	Rel-3(159.744,429.058,140.724,0)
Abs-4(5.2562,392.447,898.676,0)	Rel-4(5.2562,392.447,898.676,0)
Abs-5(610.037,749.891,965.386,0)	Rel-5(610.037,749.891,965.386,0)
Abs-6(972.232,654.206,424.028,0)	Rel-6(972.232,654.206,424.028,0)
Abs-7(484.483,250.781,769.826,0)	Rel-7(484.483,250.781,769.826,0)
Abs-8(126.599,35.5006,666.911,0)	Rel-8(126.599,35.5006,666.911,0)
Abs-9(657.245,353.739,598.14,0)	Rel-9(657.245,353.739,598.14,0)
Abs-10(304.523,664.499,14.8264,0)	Rel-10(304.523,664.499,14.8264,0)
Abs-1(780.255,705.593,924.523,0)	Rel-1(780.255,705.593,924.523,0)
Abs-0(0,0,0,0)	Rel-0(780.255,705.593,924.523,0)
Abs-2(933.405,509.883,201.967,0)	Rel-2(153.149,195.71,722.556,0)
Abs-3(159.744,429.058,140.724,0)	Rel-3(620.511,276.534,783.799,0)
Abs-4(5.2562,392.447,898.676,0)	Rel-4(774.999,313.145,25.8472,0)
Abs-5(610.037,749.891,965.386,0)	Rel-5(170.218,44.2986,40.8632,0)

## D. THE SOFTWARE INPUT DATA-SET FILE

---

Abs-6(972.232,654.206,424.028,0)	Rel-6(191.977,51.3862,500.495,0)
Abs-7(484.483,250.781,769.826,0)	Rel-7(295.772,454.812,154.698,0)
Abs-8(126.599,35.5006,666.911,0)	Rel-8(653.656,670.092,257.612,0)
Abs-9(657.245,353.739,598.14,0)	Rel-9(123.011,351.854,326.383,0)
Abs-10(304.523,664.499,14.8264,0)	Rel-10(475.732,41.0937,909.697,0)
Abs-2(933.405,509.883,201.967,0)	Rel-2(933.405,509.883,201.967,0)
Abs-0(0,0,0,0)	Rel-0(933.405,509.883,201.967,0)
Abs-1(780.255,705.593,924.523,0)	Rel-1(153.149,195.71,722.556,0)
Abs-3(159.744,429.058,140.724,0)	Rel-3(773.661,80.8246,61.243,0)
Abs-4(5.2562,392.447,898.676,0)	Rel-4(928.149,117.435,696.709,0)
Abs-5(610.037,749.891,965.386,0)	Rel-5(323.368,240.008,763.42,0)
Abs-6(972.232,654.206,424.028,0)	Rel-6(38.8273,144.324,222.061,0)
Abs-7(484.483,250.781,769.826,0)	Rel-7(448.922,259.102,567.859,0)
Abs-8(126.599,35.5006,666.911,0)	Rel-8(806.806,474.382,464.944,0)
Abs-9(657.245,353.739,598.14,0)	Rel-9(276.16,156.144,396.173,0)
Abs-10(304.523,664.499,14.8264,0)	Rel-10(628.881,154.616,187.14,0)
Abs-3(159.744,429.058,140.724,0)	Rel-3(159.744,429.058,140.724,0)
Abs-0(0,0,0,0)	Rel-0(159.744,429.058,140.724,0)
Abs-1(780.255,705.593,924.523,0)	Rel-1(620.511,276.534,783.799,0)
Abs-2(933.405,509.883,201.967,0)	Rel-2(773.661,80.8246,61.243,0)
Abs-4(5.2562,392.447,898.676,0)	Rel-4(154.488,36.6108,757.952,0)
Abs-5(610.037,749.891,965.386,0)	Rel-5(450.293,320.833,824.663,0)
Abs-6(972.232,654.206,424.028,0)	Rel-6(812.488,225.148,283.304,0)
Abs-7(484.483,250.781,769.826,0)	Rel-7(324.739,178.277,629.102,0)
Abs-8(126.599,35.5006,666.911,0)	Rel-8(33.1446,393.557,526.187,0)
Abs-9(657.245,353.739,598.14,0)	Rel-9(497.501,75.3192,457.416,0)
Abs-10(304.523,664.499,14.8264,0)	Rel-10(144.78,235.441,125.897,0)
Abs-4(5.2562,392.447,898.676,0)	Rel-4(5.2562,392.447,898.676,0)
Abs-0(0,0,0,0)	Rel-0(5.2562,392.447,898.676,0)
Abs-1(780.255,705.593,924.523,0)	Rel-1(774.999,313.145,25.8472,0)
Abs-2(933.405,509.883,201.967,0)	Rel-2(928.149,117.435,696.709,0)
Abs-3(159.744,429.058,140.724,0)	Rel-3(154.488,36.6108,757.952,0)
Abs-5(610.037,749.891,965.386,0)	Rel-5(604.781,357.444,66.7104,0)
Abs-6(972.232,654.206,424.028,0)	Rel-6(966.976,261.759,474.648,0)
Abs-7(484.483,250.781,769.826,0)	Rel-7(479.227,141.666,128.85,0)
Abs-8(126.599,35.5006,666.911,0)	Rel-8(121.343,356.947,231.765,0)
Abs-9(657.245,353.739,598.14,0)	Rel-9(651.989,38.7084,300.536,0)
Abs-10(304.523,664.499,14.8264,0)	Rel-10(299.267,272.052,883.85,0)
Abs-5(610.037,749.891,965.386,0)	Rel-5(610.037,749.891,965.386,0)
Abs-0(0,0,0,0)	Rel-0(610.037,749.891,965.386,0)
Abs-1(780.255,705.593,924.523,0)	Rel-1(170.218,44.2986,40.8632,0)
Abs-2(933.405,509.883,201.967,0)	Rel-2(323.368,240.008,763.42,0)
Abs-3(159.744,429.058,140.724,0)	Rel-3(450.293,320.833,824.663,0)
Abs-4(5.2562,392.447,898.676,0)	Rel-4(604.781,357.444,66.7104,0)
Abs-6(972.232,654.206,424.028,0)	Rel-6(362.195,95.6847,541.359,0)
Abs-7(484.483,250.781,769.826,0)	Rel-7(125.554,499.11,195.561,0)
Abs-8(126.599,35.5006,666.911,0)	Rel-8(483.438,714.391,298.476,0)
Abs-9(657.245,353.739,598.14,0)	Rel-9(47.2076,396.152,367.246,0)
Abs-10(304.523,664.499,14.8264,0)	Rel-10(305.514,85.3922,950.56,0)
Abs-6(972.232,654.206,424.028,0)	Rel-6(972.232,654.206,424.028,0)
Abs-0(0,0,0,0)	Rel-0(972.232,654.206,424.028,0)
Abs-1(780.255,705.593,924.523,0)	Rel-1(191.977,51.3862,500.495,0)
Abs-2(933.405,509.883,201.967,0)	Rel-2(38.8273,144.324,222.061,0)

---

Abs-3(159.744,429.058,140.724,0)	Rel-3(812.488,225.148,283.304,0)
Abs-4(5.2562,392.447,898.676,0)	Rel-4(966.976,261.759,474.648,0)
Abs-5(610.037,749.891,965.386,0)	Rel-5(362.195,95.6847,541.359,0)
Abs-7(484.483,250.781,769.826,0)	Rel-7(487.749,403.426,345.798,0)
Abs-8(126.599,35.5006,666.911,0)	Rel-8(845.633,618.706,242.883,0)
Abs-9(657.245,353.739,598.14,0)	Rel-9(314.987,300.468,174.112,0)
Abs-10(304.523,664.499,14.8264,0)	Rel-10(667.709,10.2925,409.201,0)
Abs-7(484.483,250.781,769.826,0)	Rel-7(484.483,250.781,769.826,0)
Abs-0(0,0,0,0)	Rel-0(484.483,250.781,769.826,0)
Abs-1(780.255,705.593,924.523,0)	Rel-1(295.772,454.812,154.698,0)
Abs-2(933.405,509.883,201.967,0)	Rel-2(448.922,259.102,567.859,0)
Abs-3(159.744,429.058,140.724,0)	Rel-3(324.739,178.277,629.102,0)
Abs-4(5.2562,392.447,898.676,0)	Rel-4(479.227,141.666,128.85,0)
Abs-5(610.037,749.891,965.386,0)	Rel-5(125.554,499.11,195.561,0)
Abs-6(972.232,654.206,424.028,0)	Rel-6(487.749,403.426,345.798,0)
Abs-8(126.599,35.5006,666.911,0)	Rel-8(357.884,215.28,102.915,0)
Abs-9(657.245,353.739,598.14,0)	Rel-9(172.762,102.958,171.685,0)
Abs-10(304.523,664.499,14.8264,0)	Rel-10(179.96,413.718,754.999,0)
Abs-8(126.599,35.5006,666.911,0)	Rel-8(126.599,35.5006,666.911,0)
Abs-0(0,0,0,0)	Rel-0(126.599,35.5006,666.911,0)
Abs-1(780.255,705.593,924.523,0)	Rel-1(653.656,670.092,257.612,0)
Abs-2(933.405,509.883,201.967,0)	Rel-2(806.806,474.382,464.944,0)
Abs-3(159.744,429.058,140.724,0)	Rel-3(33.1446,393.557,526.187,0)
Abs-4(5.2562,392.447,898.676,0)	Rel-4(121.343,356.947,231.765,0)
Abs-5(610.037,749.891,965.386,0)	Rel-5(483.438,714.391,298.476,0)
Abs-6(972.232,654.206,424.028,0)	Rel-6(845.633,618.706,242.883,0)
Abs-7(484.483,250.781,769.826,0)	Rel-7(357.884,215.28,102.915,0)
Abs-9(657.245,353.739,598.14,0)	Rel-9(530.645,318.238,68.7706,0)
Abs-10(304.523,664.499,14.8264,0)	Rel-10(177.924,628.998,652.084,0)
Abs-9(657.245,353.739,598.14,0)	Rel-9(657.245,353.739,598.14,0)
Abs-0(0,0,0,0)	Rel-0(657.245,353.739,598.14,0)
Abs-1(780.255,705.593,924.523,0)	Rel-1(123.011,351.854,326.383,0)
Abs-2(933.405,509.883,201.967,0)	Rel-2(276.16,156.144,396.173,0)
Abs-3(159.744,429.058,140.724,0)	Rel-3(497.501,75.3192,457.416,0)
Abs-4(5.2562,392.447,898.676,0)	Rel-4(651.989,38.7084,300.536,0)
Abs-5(610.037,749.891,965.386,0)	Rel-5(47.2076,396.152,367.246,0)
Abs-6(972.232,654.206,424.028,0)	Rel-6(314.987,300.468,174.112,0)
Abs-7(484.483,250.781,769.826,0)	Rel-7(172.762,102.958,171.685,0)
Abs-8(126.599,35.5006,666.911,0)	Rel-8(530.645,318.238,68.7706,0)
Abs-10(304.523,664.499,14.8264,0)	Rel-10(352.721,310.76,583.314,0)
Abs-10(304.523,664.499,14.8264,0)	Rel-10(304.523,664.499,14.8264,0)
Abs-0(0,0,0,0)	Rel-0(304.523,664.499,14.8264,0)
Abs-1(780.255,705.593,924.523,0)	Rel-1(475.732,41.0937,909.697,0)
Abs-2(933.405,509.883,201.967,0)	Rel-2(628.881,154.616,187.14,0)
Abs-3(159.744,429.058,140.724,0)	Rel-3(144.78,235.441,125.897,0)
Abs-4(5.2562,392.447,898.676,0)	Rel-4(299.267,272.052,883.85,0)
Abs-5(610.037,749.891,965.386,0)	Rel-5(305.514,85.3922,950.56,0)
Abs-6(972.232,654.206,424.028,0)	Rel-6(667.709,10.2925,409.201,0)
Abs-7(484.483,250.781,769.826,0)	Rel-7(179.96,413.718,754.999,0)
Abs-8(126.599,35.5006,666.911,0)	Rel-8(177.924,628.998,652.084,0)
Abs-9(657.245,353.739,598.14,0)	Rel-9(352.721,310.76,583.314,0)

## **D. THE SOFTWARE INPUT DATA-SET FILE**

---

## Appendix E

# Example software model results file.

The example shown here is Population best fitness, the file is shown here divided into named blocks in bold italics, these are not part of the file structure. The Statistics block is usually

### **E.1 Header block.**

```
"DS_10200_01_m010_f10_i1.00_ipc_0.10_ppc_0.10",  
RCount=10 of 10, mCount=10, pCount=1, init=99.000000,  
sys.Perturb.Std=99.000000 Normalisation Status: Normalise.UnitCube [True],  
Normalise.ZoneCount [True]
```

```
File Data [Tue Jan 25 22:19:12 2011] Compilation [Compiled - on Oct 27 2010 at 08:12:26]  
Prog Descriptor [Experiment - Multi Input File Constraint Hybrid Op RHC - EA Airspace Model -  
0.1.1.B]
```

```
FMT_AVG-[True] FMT_GRD-[False] FMT_MAX-[True] FMT_MED-[False]  
FMT_MIN-[True] FMT_SDV-[True] FMT_SUM-[False] FMT_VAR-[False]  
FMT_SEC-[False] FMT_RGP-[False]
```

### **E.2 Data Block.**

<b>Iteration</b>	<b>DVec_0</b>	<b>DVec_1</b>	<b>DVec_2</b>
0	505.639993	505.639993	505.639993
1	467.074269	471.563383	468.327769
2	453.049567	461.358732	453.276689
3	436.808900	453.090808	450.156934
4	424.923246	446.316532	440.152627
5	418.803571	442.882531	440.152627

## E. EXAMPLE SOFTWARE MODEL RESULTS FILE.

---

6	411.830716	422.764984	433.395414
7	399.611477	410.862045	426.822129
8	398.192094	410.862045	406.013065
9	396.855760	400.191417	395.788234
...			
9990	0.279473	3.115179	3.151192
9991	0.279473	3.115179	3.151192
9992	0.279473	3.115179	3.151192
9993	0.279473	3.115179	3.151192
9994	0.279473	3.115179	3.151192
9995	0.223453	3.115179	3.151192
9996	0.223453	3.098152	3.151192
9997	0.223453	3.098152	3.151192
9998	0.223453	3.098152	3.151192
9999	0.223453	3.098152	3.151192

### E.3 Statistics block.

This is another optional block in which the statistics are generated by scanning each line across all columns in the data block, thus directly extracting the data from each run at the same iteration. The number of columns in the statistics block largely reflects how many statistics have been requested in the initialisation file. The output from the range-group statistic type is the exception, this is formatted with a number of columns that reflects the number of range groups contained in the format string for that group and several other columns for various combined statistics data.

<b>Iteration</b>	<b>FMT_AVG</b>	<b>FMT_MAX</b>	<b>FMT_MIN</b>	<b>FMT_SDV</b>
0	505.639993	505.639993	505.639993	0
1	468.9884737	471.563383	468.327769	2.3163406
2	455.894996	461.358732	453.276689	4.7330967
3	446.6855473	453.090808	450.156934	8.6783077
4	437.1308017	446.316532	440.152627	11.012117
5	433.946243	442.882531	440.152627	13.184782
6	422.6637047	433.395414	422.764984	10.782705
7	412.4318837	426.822129	410.862045	13.673082
8	405.0224013	410.862045	406.013065	6.3928065
9	397.6118037	400.191417	395.788234	2.2968906
...				
9990	2.181948	3.151192	3.115179	1.6476900
9991	2.181948	3.151192	3.115179	1.6476900
9992	2.181948	3.151192	3.115179	1.6476900
9993	2.181948	3.151192	3.115179	1.6476900
9994	2.181948	3.151192	3.115179	1.6476900
9995	2.163274	3.151192	3.115179	1.6800313
9996	2.157599	3.151192	3.098152	1.6752294
9997	2.157599	3.151192	3.098152	1.6752294

---

9998	2.157599	3.151192	3.098152	1.6752294
9999	2.157599	3.151192	3.098152	1.6752294

**E. EXAMPLE SOFTWARE MODEL RESULTS FILE.**

---

## Appendix F

# RSSI input data-set neutrality test

This appendix contains the results from a set of standard Hill-climber simulation run intended to demonstrate that the randomly input datasets used in all of the simulations are not responsible for any skewing of the results. In the absence of any 'real data', (as the system that would generate this is notional), the input datasets have been generated using a software program based on the same set of virtual base classes as the main software reconstruction model. The results data is from five different input data-sets in two groups of five results plots in figures [F.1](#) & [F.2](#) for objective fitness and standard deviation results respectively, (across all ten runs), in a simulation.

The results are placed on the following page, each plot represents a composite averaged set of values from a discrete input dataset consisting of the standard problem size groupings, (PS10, PS20, PS50, PS100 and PS200 elements respectively).

## F. RSSI INPUT DATA-SET NEUTRALITY TEST

### F.1 Objective fitness plots

This group of results plots group shows the overall averaged fitness convergence plotted data from across all ten runs in a simulation. As can be seen the first, fourth & fifth plots demonstrate a particular flattening of the convergence plot from around 1000 iterations onwards, the second & third plots indicate that the convergence has more tightly contained 'turning point', again at around the 1000 iteration point.

In general the overall terminal objective fitness values are broadly similar for all input data-sets, indicating that the influence of the input data-set is not particularly strong, although there is some evidence of an effect.

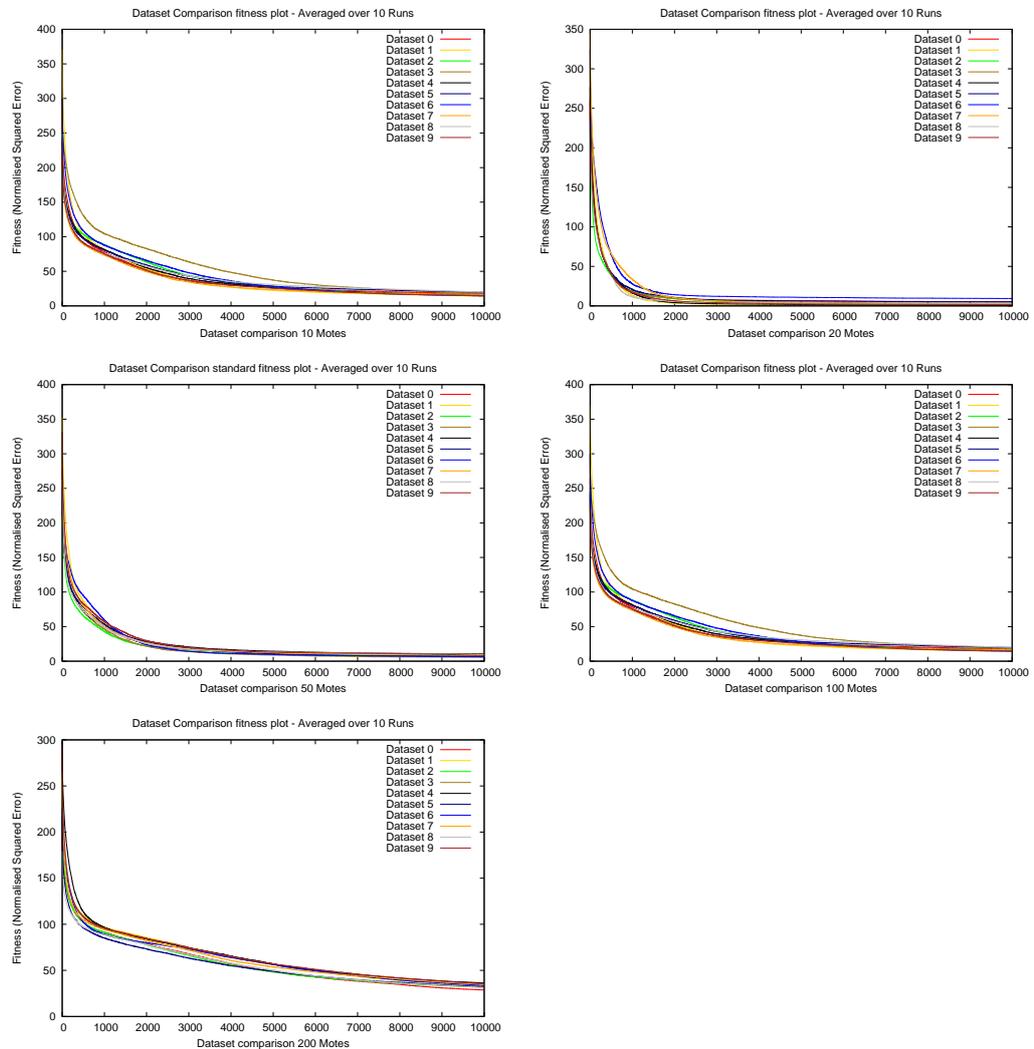


Figure F.1: Dataset comparison: 10 to 200 elements - Fitness Plots

## F.2 Standard deviation plots

This group of results plots show the overall standard deviation in the run-time results data across the ten runs and for each problem size grouping. There is clear evidence of discrepancies between the runs as would be expected in a system that is largely stochastic in operation. The standard deviation for all input data-sets shows considerable deviation through the iterations but eventually 'converges' to a range of between  $\approx 2.0$  &  $4.0$  as a terminal value.

The probable conclusion that can be drawn from this is that the input data-sets are sufficiently random and 'difficult' to solve, but that the software model ultimately gets the objective fitness values within a fairly tight range towards the end of all runs in the simulation.

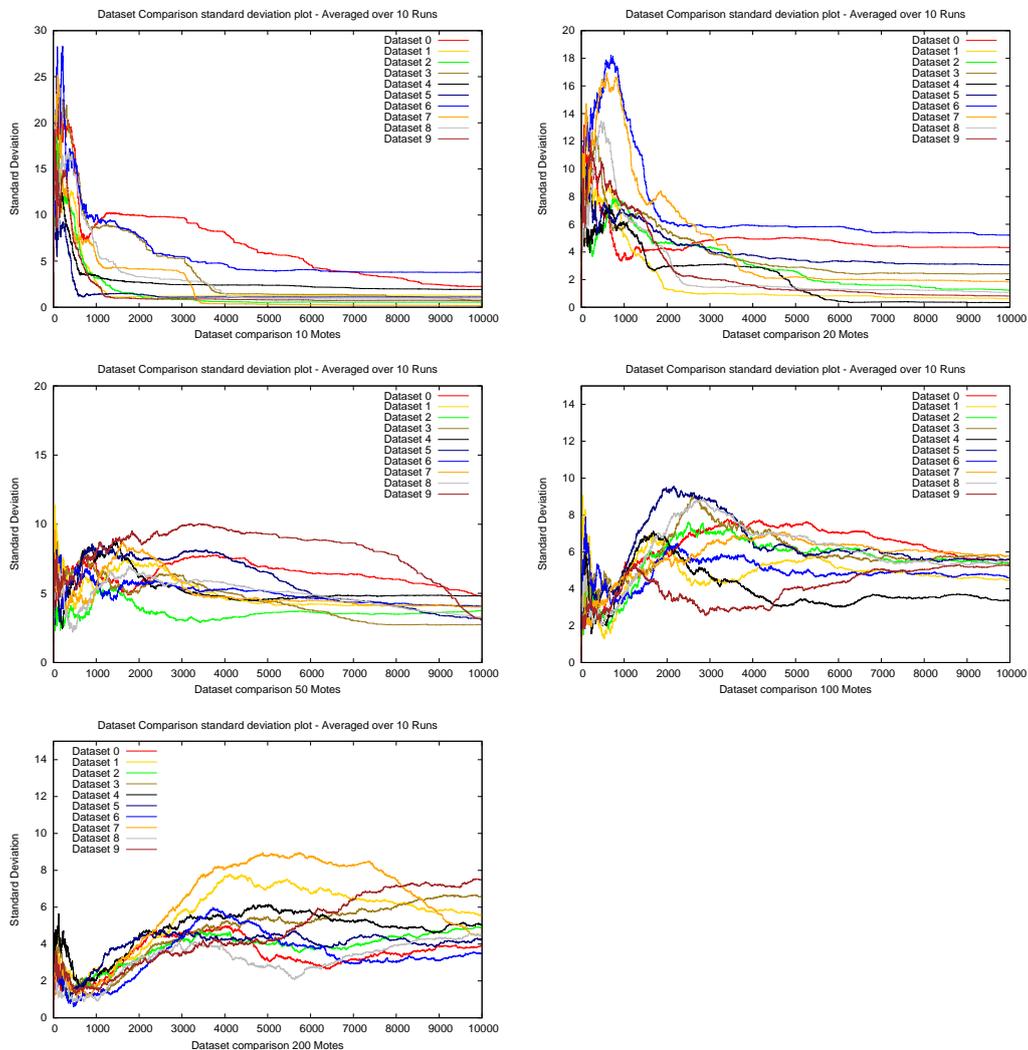


Figure F.2: Dataset comparison: 10 to 200 elements - Standard Deviation Plots

## **F. RSSI INPUT DATA-SET NEUTRALITY TEST**

---

Part VI

References



# References

- [1] AKYILDIZ, I. F., SU, W., SANKARASUBRAMANIAM, Y., AND CAYIRCI, E. A survey on sensor networks. *Communications Magazine, IEEE* 40, 8 (aug 2002), 102 – 114. [15](#), [31](#)
- [2] AKYILDIZ, I. F., SU, W., SANKARASUBRAMANIAM, Y., AND CAYIRCI, E. Wireless sensor networks: a survey. *Computer Networks* 38, 4 (2002), 393 – 422. [15](#), [31](#)
- [3] ALAVI, M., AND HENDERSON, J. C. An evolutionary strategy for implementing a decision support system. *Management Science* 27, 11 (1981), 1309–1323. [26](#)
- [4] ANTICHEVA, I., BALLINTIJN, M., BELLENOT, B., BISKUP, M., BRUN, R., BUNCIC, N., CANAL, P., CASADEI, D., COUET, O., FINE, V., FRANCO, L., GANIS, G., GHEATA, A., GONZALEZ MALINE, D., GOTO, M., IWASZKIEWICZ, J., KRESHUK, K., MARCOS SEGURA, D., MAUNDER, R., L, M., A, N., E, O., ONUCHIN, V., PANACEK, S, RADEMAKERS, F., RUSSO, P., AND TADEL, M. Root a c++ framework for petabyte data storage, statistical analysis and visualization. *Computer Physics Communications* 180, 12 (2009), 2499 – 2512. [40 YEARS OF CPC: A celebratory issue focused on quality software for high performance, grid and novel computing architectures](#); [ce:title](#). [58](#)
- [5] ASPNES, J., GOLDENBERG, D., AND YANG, Y. On the computational complexity of sensor network localization. In *Algorithmic Aspects of Wireless Sensor Networks*, S. Nikolettseas and J. Rolim, Eds., vol. 3121 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2004, pp. 32–44. 10.1007/978-3-540-27820-7\_5. [31](#)
- [6] BACK, T. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford Univ. Press, New York, NY, 1996. [23](#)
- [7] BACK, T., AND SCHWEFEL, H.-P. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation* 1, 1 (Mar. 1993), 1–23. [127](#)
- [8] BALAKRISHNAN, H., DEMAINE, E., STONEBRAKER, M., AND TELLER, S. Scalable location-aware monitoring (slam) systems. Tech. rep., Massachusetts Institute of Technology, Laboratory for Computer Science ITR, 2001. [31](#)

## REFERENCES

---

- [9] BARSOCCHI, P., LENZI, S., CHESSA, S., AND GIUNTA, G. Virtual calibration for rssi-based indoor localization with ieee 802.15.4. In *Communications, 2009. ICC '09. IEEE International Conference on* (june 2009), pp. 1–5. [18](#)
- [10] BEYER, H., AND SCHWEFEL, H. Evolution strategies: A comprehensive introduction. *Natural Computing 1* (2002), 3–52. [25](#), [127](#)
- [11] BRADLEY, P. S., AND FAYYAD, U. M. Refining initial points for k-means clustering. In *Proceedings of the Fifteenth International Conference on Machine Learning* (1998), vol. 66, San Francisco, CA, USA. [205](#)
- [12] BROWNSCOMBE, J. L., AND HALLETT, J. Experimental and field studies of precipitation particles formed by the freezing of supercooled water. *Quarterly Journal of the Royal Meteorological Society 93*, 398 (1967), 455–473. [60](#)
- [13] BULUSU, N., BYCHKOVSKIY, V., ESTRIN, D., AND HEIDEMANN, H. Scalable, ad hoc deployable rf-based localization. In *Proceedings of the Grace Hopper Conference on Celebration of Women in Computing* (2002). [31](#)
- [14] BULUSU, N., HEIDEMANN, J., AND ESTRIN, D. Gps-less low-cost outdoor localization for very small devices. *Personal Communications, IEEE 7*, 5 (oct 2000), 28–34. [36](#)
- [15] BURGARD, W., MOORS, M., FOX, D., SIMMONS, R., AND THRUN, S. Collaborative multi-robot exploration. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on* (2000), vol. 1, pp. 476–481 vol.1. [19](#)
- [16] CAMP, T., BOLENG, J., AND DAVIES, V. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing 2*, 5 (2002), 483–502. [31](#)
- [17] CHI, V. Quaternions and rotations in 3-space:. Tech. rep., CiteSeerX - Scientific Literature Digital Library and Search Engine [<http://citeseerx.ist.psu.edu/oai2>] (United States), 1998. [177](#), [206](#)
- [18] CHU, P. C., AND BEASLEY, J. E. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics 4*, 1 (June 1998), 63–86. [23](#)
- [19] CHUAN-CHIN, P., AND WAN-YOUNG, C. Mitigation of multipath fading effects to improve indoor rssi performance. *Sensors Journal, IEEE 8*, 11 (nov. 2008), 1884–1886. [17](#)
- [20] COHOON, J., MARTIN, W., AND RICHARDS, D. Genetic algorithms and punctuated equilibria in vlsi. *Parallel Problem Solving from Nature* (1991), 134–144. [24](#)

## REFERENCES

---

- [21] COLE, H. L., AND F., H. T. The driftsonde observing system development. In *13th Symposium on Meteorological Observations and Instrumentation* (19 June 2005). [14](#)
- [22] CROW, B., WIDJAJA, I., KIM, L., AND SAKAI, P. T. Ieee 802.11 wireless local area networks. *Communications Magazine, IEEE* *35*, 9 (sep 1997), 116–126. [37](#)
- [23] CUNNINGHAM, W. P., FREEMAN, D., AND MCCLOSKEY, J. F. Of radar and operations research: An appreciation of a. p. rowe (1898-1976). *Operations Research* *32*, 4 (1984), pp. 958–967. [34](#)
- [24] DAIYA, V., EBENEZER, J., MURTY, S., AND RAJ, B. Experimental analysis of rssi for distance and position estimation. In *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on* (june 2011), pp. 1093–1098. [18](#)
- [25] DARWIN, C. On the origin of species. *Soil Science* *71*, 6 (1951), 473. [23](#), [49](#)
- [26] DAVIS, W. "loran" guides pilots. *The Science News-Letter* *48* (1945), 275–276. [34](#)
- [27] DORIGO, M., AND BLUM, C. Ant colony optimization theory: A survey. *Theoretical Computer Science* *344*, 2 - 3 (2005), 243 – 278. [20](#)
- [28] DORIGO, M., AND DI CARO, G. Ant colony optimization: a new meta-heuristic. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on* (1999), vol. 2, pp. 3 vol. (xxxvii+2348). [20](#)
- [29] DORIGO, M., AND GAMBARDELLA, L. Ant colony system: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on* *1*, 1 (apr 1997), 53–66. [21](#)
- [30] DORIGO, M., AND GAMBARDELLA, L. Guest editorial special section on ant colony optimization. *IEEE Transactions on Evolutionary Computation* *6*, 4 (2002), 317. [20](#)
- [31] DRECHSLER, R., DRECHSLER, N., BECKER, B., AND ESBENSEN, H. Evolutionary algorithms in computer-aided design of integrated circuits. *World Scientific Series in Robotics and Intelligenet Systems* *18* (1999), 327–354. [24](#)
- [32] DUCKETT, T. A genetic algorithm for simultaneous localization and mapping. In *In the proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2003)* (2003), pp. 434–439. [155](#)
- [33] DUNN, S. A. The use of genetic algorithms and stochastic hill-climbing in dynamic finite element model identification. *Computers & Structures* *66*, 4 (1998), 489 – 497. [22](#)

## REFERENCES

---

- [34] DYMOND, E. G. The kew radio sonde. *Proceedings of the Physical Society* 59, 4 (1947), 645. [44](#)
- [35] EIBEN, A., HINTERDING, R., AND MICHALEWICZ, Z. Parameter control in evolutionary algorithms. *Evolutionary Computation, IEEE Transactions on* 3, 2 (jul 1999), 124–141. [127](#)
- [36] EINSTEIN, A., LAWSON, R., GEROCH, R., AND CASSIDY, D. *Relativity: The special and general theory*. Pi Pr, 1920. [33](#)
- [37] ESA. European space agency, what is galileo ? <http://www.esa.int/esaNA/galileo.html>. [36](#)
- [38] FANG, Z., ZHAO, Z., GENG, D., XUAN, Y., DU, L., AND CUI, X. Rssi variability characterization and calibration method in wireless sensor network. In *Information and Automation (ICIA), 2010 IEEE International Conference on* (june 2010), pp. 1532–1537. [17](#)
- [39] FENG, J., KOUSHANFAR, F., AND POTKONJAK, M. System-architectures for sensor networks issues, alternatives, and directions. In *Computer Design: VLSI in Computers and Processors, 2002. Proceedings. 2002 IEEE International Conference on* (2002), pp. 226–231. [15](#)
- [40] FIERRO, R., SONG, P., DAS, A., AND KUMAR, V. Cooperative control of robot formations. In *Cooperative Control and Optimization: Series on Applied Optimization* (2002), R. Murphey and P. P. E. N. York, Eds., Kluwer Academic Press - Springer, pp. 79–93. [19](#)
- [41] FUJIWARA, Y., AND SAWAI, H. Evolutionary computation applied to mesh optimization of a 3-d facial image. *Evolutionary Computation, IEEE Transactions on* 3, 2 (jul 1999), 113–123. [23](#), [155](#)
- [42] GAGNE, C., AND PARIZEAU, M. Open beagle: A new versatile c++ framework for evolutionary computations. In *In: Late-Breaking Papers of the 2002 Genetic and Evolutionary Computation Conference (GECCO 2002)* (2002), pp. 161–168. [58](#)
- [43] GARCIA-HERNANDEZ, C. F., GONZALEZ P. H, I., HERNANDEZ J, G., AND DIAZ J. A., P. Wireless sensor networks and applications: a survey. *International Journal of Computer Science and Network Security VOL.7 No.3* (March 2007). [15](#)
- [44] GAREY, M. R., AND JOHNSON, D. S. Computers and intractability: A guide to the theory of np-completeness. In *Computers and intractability: A guide to the theory of NP-completeness* (W. H. Freeman and Company, San Francisco, 1980), M. P. Drazin, Ed., vol. 3, bulletin of the American Mathematical. Society. (N.S.), pp. xii + 338. [6](#), [57](#)

- 
- [45] GLOVER, F. Tabu search - part i. *ORSA Journal on computing* 1, 3 (1989), 190–206. [22](#)
- [46] GLOVER, F. Tabu search - part ii. *ORSA Journal on computing* 2, 1 (1990), 4–32. [22](#)
- [47] GLOVER, F. Tabu search: A tutorial. *Interfaces* 20, 4 (1990), 74–94. [22](#)
- [48] GLOVER, F., TAILLARD, E., AND TAILLARD, E. A user’s guide to tabu search. *Annals of operations research* 41, 1 (1993), 1–28. [22](#)
- [49] GOLDBERG, D. E. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989. [23](#), [155](#)
- [50] GORNITZ, V. The great ice meltdown and rising seas: Lessons for tomorrow. [http://www.giss.nasa.gov/research/briefs/gornitz\\_10/](http://www.giss.nasa.gov/research/briefs/gornitz_10/). [192](#)
- [51] GUNES, M., SORGES, U., AND BOUAZIZI, I. Ara-the ant-colony based routing algorithm for manets. In *Parallel Processing Workshops, 2002. Proceedings. International Conference on* (2002), pp. 79 – 85. [21](#)
- [52] GUO, Y., AND PARKER, L. A distributed and optimal motion planning approach for multiple mobile robots. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on* (2002), vol. 3, pp. 2612 –2619. [19](#)
- [53] GUSTAFSSON, F., AND GUNNARSSON, F. Positioning using time-difference of arrival measurements. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on* (april 2003), vol. 6, pp. VI – 553–6 vol.6. [34](#)
- [54] HANSEN, N., AND OSTERMEIER, A. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computing* 9, 2 (2001), 159–195. [25](#), [108](#)
- [55] HART, J. C., FRANCIS, G. K., AND KAUFFMAN, L. H. Visualizing quaternion rotation. *ACM Trans. Graph.* 13, 3 (July 1994), 256–276. [177](#), [206](#)
- [56] HEIDEMANN, J., AND BULUSU, N. Using geospatial information in sensor networks. Tech. rep., University of California eScholarship Repository [<http://repositories.cdlib.org/cgi/oai2.cgi>] (United States), 2001. [36](#)
- [57] HENDERSON, P. *Object-Oriented Specification and Design with C++*. McGraw-Hill, 1993. Address: New York. [28](#), [52](#)
- [58] HEURTEFEUX, K., AND VALOIS, F. Is rssi a good choice for localization in wireless sensor network? In *Advanced Information Networking and Applications (AINA), 2012 IEEE 26th International Conference on* (march 2012), pp. 732 –739. [16](#)

## REFERENCES

---

- [59] HO, S., SHU, L., AND CHEN, J. Intelligent evolutionary algorithms for large parameter optimization problems. *Evolutionary Computation, IEEE Transactions on* 8, 6 (dec. 2004), 522 – 541. 127
- [60] HOLIN, H. *The Quaternionic Exponential (and beyond)*, 1999. 177, 206
- [61] HOLIN, H. quaternion.pdf. pdf, 2003. 177, 206
- [62] HOROWITZ, E., AND SAHNI, S. Computing partitions with applications to the knapsack problem. *J. ACM* 21, 2 (Apr. 1974), 277–292. 23
- [63] HSIEH, M., KUMAR, V., AND TAYLOR, C. Constructing radio signal strength maps with multiple robots. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on* (26-may 1, 2004), vol. 4, pp. 4184 – 4189 Vol.4. 19, 37
- [64] HU, L., AND EVANS, D. Localization for mobile sensor networks. In *Proceedings of the 10th annual international conference on Mobile computing and networking* (New York, NY, USA, 2004), MobiCom '04, ACM, pp. 45–57. 31
- [65] HUANG, X. Antenna polarization as complementarities on rssi based location identification. In *Wireless Pervasive Computing, 2009. ISWPC 2009. 4th International Symposium on* (feb. 2009), pp. 1 –5. 17
- [66] IPCC. Intergovernmental panel of climate change. <http://www.ipcc.ch/>. 192
- [67] JAATINEN, J., AND ELMS, J. B. On the windfinding of loran-c, gps and windfinding radar, 1999. 46
- [68] JILLA, C. *A multiobjective, multidisciplinary design optimization methodology for the conceptual design of distributed satellite systems*. PhD thesis, Massachusetts Institute of Technology, 2002. 23
- [69] JOUZEL, J., MERLIVAT, L., AND ROTH, E. Isotopic study of hail. *J. Geophys. Res.* 80, 36 (1975), 5015–5030. 60
- [70] KALYANMOY, D. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley and Sons, Inc., 2001. 155
- [71] KENNEDY, J., AND EBERHART, R. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on* (nov/dec 1995), vol. 4, pp. 1942 –1948 vol.4. 21
- [72] KHEMAPECH, I., DUNCAN, I., AND MILLER, A. A survey of wireless sensor networks technology. In *6th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting* (2005). 15
- [73] KLEINBERG, J., AND TARDOS, E. Np and computational intractability. pdf, 2008. 6, 57

## REFERENCES

---

- [74] KOZIEL, S., AND MICHALEWICZ, Z. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation* 7, 1 (Mar. 1999), 19–44. [127](#)
- [75] KOZIEL, S., AND SZCZESNIAK, W. Application of evolutionary algorithms-to vlsi circuits partitioning with reduction of thermal interactions between elements. In *THERMINIC: international workshop on thermal investigations of ICs and microstructures* (1999), pp. 354–359. [24](#)
- [76] KOZIEL, S., AND SZCZESNIAK, W. Reducing average and peak temperatures of vlsi cmos circuits by means of evolutionary algorithm applied to high level synthesis. *Microelectronics Journal* 34, 12 (2003), 1167 – 1174. [Thermal Investigations of integrated circuits and systems at Therminic 2002](#). [24](#)
- [77] KRISHNA, K., AND NARASIMHA MURTY, M. Genetic k-means algorithm. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 29, 3 (jun 1999), 433 –439. [205](#)
- [78] KUIPERS, J. B. Quaternion and rotation sequences. In *Geometry, Integrability and Quantization* (September 2000), I. M. Mladenov and G. L. Naber, Eds., Coral Press, pp. 127–143. [177](#), [206](#)
- [79] LI, H., JIAO, Y., ZHANG, L., AND GU, Z. Genetic algorithm based on the orthogonal design for multidimensional knapsack problems. In *Proceedings of the Second international conference on Advances in Natural Computation - Volume Part I* (Berlin, Heidelberg, 2006), ICNC’06, Springer-Verlag, pp. 696–705. [23](#), [73](#)
- [80] LIENIG, J. A parallel genetic algorithm for performance-driven vlsi routing. *Evolutionary Computation, IEEE Transactions on* 1, 1 (apr 1997), 29 –39. [24](#)
- [81] LIENIG, J. Physical design of vlsi circuits and the application of genetic algorithms. Tech. rep., CiteSeerX - Scientific Literature Digital Library and Search Engine [<http://citeseerx.ist.psu.edu/oai2>] (United States), 1997. [24](#)
- [82] LYMBEROPOULOS, D., LINDSEY, Q., AND SAVVIDES, A. An empirical characterization of radio signal strength variability in 3-d iee 802.15.4 networks using monopole antennas. In *Wireless Sensor Networks*, K. Rmer, H. Karl, and F. Mattern, Eds., vol. 3868 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006, pp. 326–341. [10.1007/11669463\\_24](#). [39](#)
- [83] MANJOO, F. Dust keeping the lights off. <http://www.wired.com/news/technology/0,1282,44101,00.html>, May 2001. [15](#)
- [84] MESTER, D., RONIN, Y., MINKOV, D., NEVO, E., AND KOROL, A. Constructing large-scale genetic maps using an evolutionary strategy algorithm. *Genetics* 165, 4 (2003), 2269–2282. [25](#), [73](#)

## REFERENCES

---

- [85] MICHALEWICZ, Z., AND SCHOENAUER, M. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation* 4, 1 (Mar. 1996), 1–32. [127](#)
- [86] MOORE, G. Cramming more components onto integrated circuits. *Proceedings of the IEEE* 86, 1 (1998), 82–85. [24](#)
- [87] NOAA. National oceanic and atmospheric administration, climate prediction center. <http://www.cpc.ncep.noaa.gov/>. [192](#)
- [88] NOAA. National oceanic and atmospheric administration, hail size chart. [http://www.erh.noaa.gov/aly/Severe/HailSize\\_Chart.htm](http://www.erh.noaa.gov/aly/Severe/HailSize_Chart.htm). [60](#)
- [89] OED. *The Oxford English Dictionary, A Dictionary of Psychology*. Oxford University Press. Oxford Reference Online, 2009. [28](#), [29](#), [30](#)
- [90] OED. *The Oxford English Dictionary, Concise Medical Dictionary*. Oxford University Press, Oxford Reference Online, 2010. [29](#)
- [91] OED. *The Oxford English Dictionary*. Oxford University Press, Oxford Reference Online, 2011. [28](#)
- [92] OPENGL. *Open GL Documentation*. Open GL, 2012. [206](#)
- [93] ORIGLIO, V., AND WEISSTEIN, E. Googol from mathworld—a wolfram web resource. <http://mathworld.wolfram.com/Googol.html>, 2012. [72](#)
- [94] PATWARI, N., AND HERO III, A. O. Using proximity and quantized rss for sensor localization in wireless networks, 2003. [39](#), [202](#)
- [95] PEREIRA-NETO, A., UNSIHUAY, C., AND SAAVEDRA, O. Efficient evolutionary strategy optimisation procedure to solve the nonconvex economic dispatch problem with generator constraints. *Generation, Transmission and Distribution, IEE Proceedings- 152*, 5 (sept. 2005), 653 – 660. [25](#), [127](#)
- [96] PIRES, R., WANNER, L., AND FROHLICH, A. An efficient calibration method for rssi-based location algorithms. In *Industrial Informatics, 2008. INDIN 2008. 6th IEEE International Conference on* (july 2008), pp. 1183 –1188. [19](#)
- [97] POTDAR, V., SHARIF, A., AND CHANG, E. Wireless sensor networks: A survey. In *Advanced Information Networking and Applications Workshops, 2009. WAINA '09. International Conference on* (may 2009), pp. 636 –641. [15](#)
- [98] RAMIRO, V., VELA, C. R., PUENTE, J., AND GOMEZ, A. A knowledge-based evolutionary strategy for scheduling problems with bottlenecks. *European Journal of Operational Research* 145, 1 (2003), 57 – 71. [25](#), [127](#)

- 
- [99] REDELSPERGER, J.-L., THORNCROFT, C. D., DIEDHIU, A., LEBEL, T., PARKER, D. J., AND POLCHER, J. African monsoon multidisciplinary analysis: An international research project and field campaign. *Bull. Amer. Meteor. Soc.* 87, 12 (Dec. 2006), 1739–1746. [14](#), [48](#)
- [100] REGHELIN, R., AND FRÖHLICH, A. A. A decentralized location system for sensor networks using cooperative calibration and heuristics. In *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems* (New York, NY, USA, 2006), MSWiM '06, ACM, pp. 139–146. [19](#)
- [101] REICHENBACH, F., AND TIMMERMANN, D. Indoor localization with low complexity in wireless sensor networks. In *Industrial Informatics, 2006 IEEE International Conference on* (aug. 2006), pp. 1018 –1023. [18](#)
- [102] ROBERTS, J. J., BEST, B. D., DUNN, D. C., TREML, E. A., AND HALPIN, P. Marine geospatial ecology tools: An integrated framework for ecological geospatial processing with arcgis, python, r, matlab, and c++. *Environmental Modelling & Software* 25, 10 (2010), 1197 – 1207. [58](#)
- [103] ROLLINGS, G., AND CORNE, D. Intelligent operators for localisation of dynamic smart dust networks. In *Eighth International Conference on Hybrid Intelligent Systems, 2008. HIS '08.* (2008), pp. 477 – 482. [ii](#)
- [104] ROLLINGS, G., AND CORNE, D. Relative distance identification in "smart dust" networks for environmental modelling. In *AIKED'08 Proceedings of the 7th WSEAS International Conference on Artificial intelligence, knowledge engineering and data bases* (2008). [ii](#)
- [105] RÖMER, K. Time and location in sensor networks. <http://people.inf.ethz.ch/roemer/papers/spacetime-kuvs03.pdf>, 2003. [36](#)
- [106] ROSS, K., AND TSANG, D. The stochastic knapsack problem. *Communications, IEEE Transactions on* 37, 7 (jul 1989), 740 –747. [23](#)
- [107] ROUBOS, J., VAN STRATEN, G., AND VAN BOXTEL, A. An evolutionary strategy for fed-batch bioreactor optimization; concepts and performance. *Journal of Biotechnology* 67, 23 (1999), 173 – 187. [25](#)
- [108] ROUMELIOTIS, S., AND BEKEY, G. Distributed multirobot localization. *Robotics and Automation, IEEE Transactions on* 18, 5 (oct 2002), 781 – 795. [19](#)
- [109] SALLAI, J., BALOGH, G., MARTI, M., LDECZI, A., AND KUSY, B. Acoustic ranging in resource-constrained sensor networks. In *In Proceedings of ICWN '04 (Las Vegas, Nv)* (2004), CSREA Press, p. 04. [34](#)
- [110] SHAHOOKAR, K., AND MAZUMDER, P. Vlsi cell placement techniques. *ACM Comput. Surv.* 23, 2 (June 1991), 143–220. [24](#)

## REFERENCES

---

- [111] SHI, Y., AND EBERHART, R. Empirical study of particle swarm optimization. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on* (1999), vol. 3, pp. 3 vol. (xxxvii+2348). 21
- [112] SICHITIU, M., AND RAMADURAI, V. Localization of wireless sensor networks with a mobile beacon. In *Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on* (oct. 2004), pp. 174 – 183. 31
- [113] SURVEY, C. A. 2011 survey - a unique scientific expedition across the arctic ocean. <http://www.catlinarcticsurvey.com/2011-mission/>, 2011. 192
- [114] THORNTON, K. libsequence: a c++ class library for evolutionary genetic analysis. *Bioinformatics (Oxford, England) 19* (2003), 2325–2327. 58
- [115] THORPEX. The observing system research and predictability experiment. [http://www.wmo.int/pages/prog/arep/wwrp/new/thorpex\\_new.html](http://www.wmo.int/pages/prog/arep/wwrp/new/thorpex_new.html). 14, 48
- [116] TUSON, A. Optimisation with hillclimbing on steroids: an overview of neighbourhood search techniques. *DAI RESEARCH PAPER* (1998). 22
- [117] UCAR. University corporation for atmospheric research. Web page. 14
- [118] UCAR. University corporation for atmospheric research., the drift sonde. <http://www.eol.ucar.edu/instrumentation/sounding/driftsonde>. 14, 47
- [119] UKMO. The uk meteorological office, a look at all aspects of climate, climate science and climate change. <http://www.metoffice.gov.uk/climate-change>. 192
- [120] UKMO. The uk meteorological office., synoptic weather charts. [http://www.metoffice.gov.uk/aviation/samples/subs\\_synoptic.html](http://www.metoffice.gov.uk/aviation/samples/subs_synoptic.html). 46
- [121] UKMO. The uk meteorological office, fact sheet no. 13: Upper air observations and the tephigram. *The UK Meteorological Office* (2007). Crown Copyright Updated July 2007 Met Office National Meteorological Library and Archive. 44
- [122] UNFCCC. United nations framework convention on climate change, doha climate change conference - november 2012. <http://unfccc.int/2860.php>, November 2012. 192
- [123] VAISALA. Vaisala radiosonde rs92-d. <http://www.vaisala.com/Vaisala%20Documents/Brochures/Datasheets/RD94-Dropsonde-Datasheet-B210936EN-A-LoRes.pdf>, 2010. 43, 45
- [124] VAISALA. Vaisala rd94 - dropsonde datasheet. <http://www.vaisala.com/Vaisala%20Documents/Brochures%20and%20Datasheets/RD94-Dropsonde-Datasheet-B210936EN-A-LoRes.pdf>, 2010. 46
- [125] VAN DER MERWE, D., AND ENGELBRECHT, A. Data clustering using particle swarm optimization. In *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on* (dec. 2003), vol. 1, pp. 215 – 220 Vol.1. 21

- 
- [126] VESTERSTROM, J., AND THOMSEN, R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Evolutionary Computation, 2004. CEC2004. Congress on* (june 2004), vol. 2, pp. 1980 – 1987 Vol.2. [127](#)
- [127] VICCI, L. Quaternions and rotations in 3-space: The algebra and its geometric interpretation, 2001. [177](#), [206](#)
- [128] VIDYARTHI, G., NGOM, A., AND STOJMENOVIC, I. A hybrid channel assignment approach using an efficient evolutionary strategy in wireless mobile networks. *Vehicular Technology, IEEE Transactions on* 54, 5 (sept. 2005), 1887 – 1895. [25](#)
- [129] VIEIRA, M., COELHO, C.N., J., DA SILVA, D.C., J., AND DA MATA, J. Survey on wireless sensor network devices. In *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference* (sept. 2003), vol. 1, pp. 537 – 544 vol.1. [14](#)
- [130] WADHWA, M., SONG, M., RALI, V., AND SHETTY, S. The impact of antenna orientation on wireless sensor network performance. In *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on* (aug. 2009), pp. 143 –147. [18](#)
- [131] WAGSTAFF, K., CARDIE, C., ROGERS, S., AND SCHROEDL, S. Constrained k-means clustering with background knowledge. In *In ICML (2001)*, Morgan Kaufmann, pp. 577–584. [205](#)
- [132] WALL, M. Galib: A c++ library of genetic algorithm components. *Mechanical Engineering Department, Massachusetts Institute of Technology* 87 (1996). [58](#)
- [133] WAN, C.-Y., EISENMAN, S. B., AND CAMPBELL, A. T. Coda: congestion detection and avoidance in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems* (New York, NY, USA, 2003), SenSys '03, ACM, pp. 266–279. [15](#)
- [134] WANG, C., AND CHANG, J. S. A three-dimensional numerical model of cloud dynamics, microphysics, and chemistry 3. redistribution of pollutants. *J. Geophys. Res.* 98, D9 (1993), 16787–16798. [60](#)
- [135] WANG, H., WANG, D., AND YANG, S. A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems. *Soft Computing - A Fusion of Foundations, Methodologies and Applications* 13 (2009), 763–780. [10.1007/s00500-008-0347-3](#). [22](#)
- [136] WHEELER, M. D., AND IKEUCHI, K. Iterative estimation of rotation and translation using the quaternion, 1995. [177](#), [206](#)

## REFERENCES

---

- [137] WHITEHOUSE, K., JIANG, F., KARLOF, C., WOO, A., AND CULLER, D. Sensor field localization: A deployment and empirical analysis. Tech. Rep. UCB/CSD-04-1349, EECS Department, University of California, Berkeley, Apr 2004. [202](#)
- [138] WHITMAN, W. B. COLEMAN, D. C., AND WIEBE, W. J. Prokaryotes: The unseen majority. *Proceedings - National Academy of Sciences, USA Vol 95; Number 12* (1998), pages 6578–6583. [71](#)
- [139] WILLIAMS, E., CROSSLEY, W., AND LANG, T. Average and maximum revisit time trade studies for satellite constellations using a multiobjective genetic algorithm. *The Journal of the astronautical sciences* *49*, 3 (2001), 385–400. [23](#)
- [140] WINOGRAD, J., AND HAMID NAWAB, S. A c++ software environment for the development of embedded signal processing systems. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on* (may 1995), vol. 4, pp. 2715 –2718 vol.4. [58](#)
- [141] WU, H., LEE, Y., TSENG, H., JAN, Y., AND CHUANG, M. Study of characteristics of rssi signal. In *Industrial Technology, 2008. ICIT 2008. IEEE International Conference on* (april 2008), pp. 1 –3. [17](#)
- [142] XIANG, Z., ZHANG, H., HUANG, J., SONG, S., AND ALMERTH, K. A hidden environment model for constructing indoor radio maps. In *World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a* (june 2005), pp. 395 – 400. [19](#)
- [143] ZHENG, J., WU, C., CHU, H., AND JI, P. Localization algorithm based on rssi and distance geometry constrain for wireless sensor network. In *Electrical and Control Engineering (ICECE), 2010 International Conference on* (june 2010), pp. 2836 –2839. [18](#)
- [144] ZITZLER, E., KALYANMOY, D., AND THIELE, L. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation* *8*, 2 (June 2000), 173–195. [155](#)