



A FRAMEWORK FOR EVOLUTIONARY
OPTIMIZATION APPLICATIONS IN WATER
DISTRIBUTION SYSTEMS

*Submitted by Mark Stephen Morley, to the University of Exeter as a thesis for
the degree of Doctor of Philosophy in Engineering, March 2008.*

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.

..... (signature)

Abstract

The application of optimization to Water Distribution Systems encompasses the use of computer-based techniques to problems of many different areas of system design, maintenance and operational management. As well as laying out the configuration of new WDS networks, optimization is commonly needed to assist in the rehabilitation or reinforcement of existing network infrastructure in which alternative scenarios driven by investment constraints and hydraulic performance are used to demonstrate a cost-benefit relationship between different network intervention strategies. Moreover, the ongoing operation of a WDS is also subject to optimization, particularly with respect to the minimization of energy costs associated with pumping and storage and the calibration of hydraulic network models to match observed field data.

Increasingly, Evolutionary Optimization techniques, of which Genetic Algorithms are the best-known examples, are applied to aid practitioners in these facets of design, management and operation of water distribution networks as part of Decision Support Systems (DSS). Evolutionary Optimization employs processes akin to those of natural selection and “survival of the fittest” to manipulate a population of individual solutions, which, over time, “evolve” towards optimal solutions. Such algorithms are characterized, however, by large numbers of function evaluations. This, coupled with the computational complexity associated with the hydraulic simulation of water networks incurs significant computational overheads, can limit the applicability and scalability of this technology in this domain.

Accordingly, this thesis presents a methodology for applying Genetic Algorithms to Water Distribution Systems. A number of new procedures are presented for improving the performance of such algorithms when applied to complex engineering problems. These techniques approach the problem of minimising the impact of the inherent computational complexity of these problems from a number of angles. A novel genetic representation is presented which combines the algorithmic simplicity of the classical binary string of the Genetic Algorithm with the performance advantages inherent in an integer-based representation. Further algorithmic improvements are demonstrated with an intelligent mutation operator that “learns” which genes have the greatest impact on the quality of a solution and concentrates the mutation operations on those genes. A technique for

implementing caching of solutions – recalling the results for solutions that have already been calculated - is demonstrated to reduce runtimes for Genetic Algorithms where applied to problems with significant computation complexity in their evaluation functions. A novel reformulation of the Genetic Algorithm for implementing robust stochastic optimizations is presented which employs the caching technology developed to produce an multiple-objective optimization methodology that demonstrates dramatically improved quality of solutions for given runtime of the algorithm.

These extensions to the Genetic Algorithm techniques are coupled with a supporting software library that represents a standardized modelling architecture for the representation of connected networks. This library gives rise to a system for distributing the computational load of hydraulic simulations across a network of computers. This methodology is established to provide a viable, scalable technique for accelerating evolutionary optimization applications.

Keywords

Evolutionary Optimization, Genetic Algorithms, Hydroinformatics, Caching, Multiple-Objective Optimization, Distributed Computing.

Acknowledgements

I am deeply grateful to my family for their enduring support over the period that I have been working towards this thesis. Their consistently positive attitude has ensured that this thesis has finally come to fruition.

In addition, I would like to thank my supervisor, Professor Dragan Savić, not least for his forbearance and perseverance in the face of persistent obduracy against completing this thesis, but also for (twice) rescuing me from the wilds of the commercial sector and introducing me to the academic environment at the Centre for Water Systems in Exeter.

Foremost amongst my (erstwhile) colleagues I would like to express my appreciation to Mr. Roger Atkinson and, latterly, Dr. Carla Tricarico for attempting to instil in me the belief that this thesis could be completed in a meaningful fashion – and that doing so was worthwhile. I would also like to thank Mr. Josef Bicik, Drs. Zoran Kapelan, Ed Keedwell, Francesco di Pierro and Darko Joksimović for their help and support along with all my other colleagues and friends that I have met over *so* many years at the Centre.

Finally, to my two sons, Alexander and Christopher, to whom this thesis is dedicated: without them, it would surely have been completed much sooner 😊

Mark S Morley

Exeter

March 2008

Table of Contents

Abstract	3
Keywords	4
Acknowledgements	5
Table of Contents	7
List of Tables	17
List of Figures	21
Glossary	27
Definitions	27
List of Abbreviations	31
Chapter 1. Introduction	35
1.1. Background.....	35
1.2. Aims of Research.....	36
1.3. Objectives	36
1.4. Thesis Structure	37
Chapter 2. Optimization in Water Distribution Systems	39
2.1. Literature Review.....	39
2.1.1. Genetic Algorithms	39
2.1.1.1. <i>Pump Optimization</i>	39
2.1.1.2. <i>Network Design and Rehabilitation</i>	40
2.1.1.3. <i>Network Calibration</i>	44
2.1.1.4. <i>Water Quality Optimization</i>	45
2.1.1.5. <i>Accommodating uncertainty in GAs</i>	45
2.1.2. Other Optimization Techniques.....	46
2.1.2.1. <i>Linear Programming</i>	46
2.1.2.2. <i>Heuristic Approaches</i>	46
2.1.2.3. <i>Cellular Automata</i>	47
2.1.2.4. <i>Particle Swarm Optimization</i>	47

2.1.2.5. <i>Simulated Annealing</i>	48
2.1.2.6. <i>Ant Colony Simulation</i>	49
2.2. Summary.....	49
Chapter 3. Genetic Algorithms	51
3.1. Introduction.....	51
3.2. Methodology.....	52
3.2.1. Algorithm operation.....	52
3.2.1.1. <i>Algorithm types</i>	54
3.2.1.2. <i>Selection</i>	55
3.2.1.3. <i>Recombination</i>	56
3.2.1.4. <i>Mutation</i>	57
3.2.1.5. <i>Replacement</i>	57
3.2.2. Solution Representation.....	58
3.2.2.1. <i>Genotype Representation (Encoding)</i>	59
3.2.2.2. <i>Decoding</i>	62
3.2.2.3. <i>Evaluation</i>	62
3.3. Implementation.....	62
3.3.1. Algorithm Modularity.....	64
3.3.2. Genetic Representation.....	67
3.3.2.1. <i>Chromosome</i>	67
3.3.2.2. <i>Genome</i>	67
3.3.3. Third Party Extensions.....	68
3.4. Conclusions.....	68
Chapter 4. Extending the GA methodology	73
4.1. Introduction.....	73
4.2. Binary String Implementation.....	73
4.2.1. Introduction.....	73
4.2.1.1. <i>Genotype Representations</i>	73
4.2.2. Conventional Representations.....	77
4.2.3. Hybridized integer gene.....	78
4.2.3.1. <i>Crossover representation</i>	79

4.2.3.2. Mutation representation.....	81
4.2.4. Experimentation.....	81
4.2.5. Conclusions.....	82
4.3. Binary String Caching.....	83
4.3.1. Introduction.....	83
4.3.2. Implementation.....	83
4.3.3. Experimentation.....	83
4.3.4. Conclusions.....	84
4.4. Solution Caching.....	85
4.4.1. Red-Black Binary tree cache.....	87
4.4.1.1. Multi-tier cache.....	90
4.4.2. Judy Tree Cache.....	91
4.4.2.1. Example.....	95
4.4.3. Experimentation.....	97
4.4.4. Conclusions.....	103
4.5. Non-Repeating GA (NRGA).....	104
4.6. Adaptive Differential Mutation.....	106
4.6.1. Introduction.....	106
4.6.1.1. Sensitivity and Trend Score Implementation.....	107
4.6.2. Differential Mutation Implementation.....	108
4.6.3. Cellular Automaton Mutation Implementation.....	109
4.6.4. Conclusions.....	110
4.7. Conclusions.....	111
Chapter 5. Distributed Evaluation for EPANET: deEPANET.....	113
5.1. Introduction.....	113
5.1.1. Parallelization of Optimization.....	114
5.2. Implementation.....	116
5.2.1. Robust networking.....	118
5.2.2. Advanced processor architectures.....	118

5.2.3.	Cross platform characteristics	119
5.3.	Application	119
5.4.	Case Study Network	120
5.5.	Results	122
5.6.	Distributing Stochastic Computation	126
5.7.	Conclusions	130
Chapter 6.	Single Objective Optimization Problems	131
6.1.	Introduction	131
6.1.1.	Genetic Representation	132
6.1.2.	Heterozygous Chromosomes	132
6.1.3.	Caching	133
6.1.4.	Adaptive Differential Mutation	134
6.1.5.	Distributed Performance	135
6.2.	New York Tunnels	135
6.2.1.	Problem Formulation	135
6.2.2.	Network Configuration	136
6.2.3.	GA Configuration	138
6.2.4.	Genetic Representation	139
6.2.4.1.	<i>Binary String</i>	139
6.2.4.2.	<i>Gray Binary String</i>	139
6.2.4.3.	<i>Integer</i>	140
6.2.4.4.	<i>Hybrid Integer</i>	140
6.2.4.5.	<i>Comparative Analysis</i>	141
6.2.4.6.	<i>Runtime Performance</i>	143
6.2.5.	Heterozygous Chromosomes	143
6.2.5.1.	<i>Binary String</i>	143
6.2.5.2.	<i>Integer</i>	144
6.2.5.3.	<i>Hybrid Integer</i>	145
6.2.5.4.	<i>Comparative Analysis</i>	146
6.2.5.5.	<i>Runtime Performance</i>	147
6.2.6.	Caching	148

6.2.7.	Adaptive Differential Mutation	148
6.2.8.	Distributed Performance	149
6.3.	Hanoi	151
6.3.1.	Problem Formulation.....	151
6.3.2.	Network Configuration.....	151
6.3.3.	GA Configuration.....	154
6.3.4.	Genetic Representation.....	155
6.3.4.1.	<i>Binary String</i>	155
6.3.4.2.	<i>Integer</i>	155
6.3.4.3.	<i>Hybrid Integer</i>	156
6.3.4.4.	<i>Comparative Analysis</i>	156
6.3.4.5.	<i>Runtime Performance</i>	157
6.3.5.	Caching	158
6.3.6.	Adaptive Differential Mutation	158
6.3.7.	Distributed Performance	159
6.3.8.	Optimal Solution Details	160
6.4.	Piedemonte San Germano.....	161
6.4.1.	Problem Formulation.....	161
6.4.2.	Network Configuration.....	162
6.4.3.	GA Configuration.....	165
6.4.4.	Genetic Representation.....	165
6.4.4.1.	<i>Binary String</i>	165
6.4.4.2.	<i>Integer</i>	166
6.4.4.3.	<i>Hybrid Integer</i>	167
6.4.4.4.	<i>Comparative Analysis</i>	167
6.4.4.5.	<i>Runtime Performance</i>	168
6.4.5.	Heterozygous Chromosomes	169
6.4.5.1.	<i>Binary String</i>	169
6.4.5.2.	<i>Integer</i>	170
6.4.5.3.	<i>Hybrid Integer</i>	171
6.4.5.4.	<i>Comparative Analysis</i>	172
6.4.5.5.	<i>Runtime Performance</i>	174

6.4.6.	Caching.....	174
6.4.7.	Adaptive Differential Mutation.....	174
6.4.8.	Distributed Performance.....	175
6.5.	Conclusions.....	177
Chapter 7.	Multiple Objective Optimization Problems	180
7.1.	Introduction.....	180
7.2.	New York Tunnels	182
7.2.1.	Genetic Representation.....	182
7.2.1.1.	<i>Binary String</i>	182
7.2.1.2.	<i>Integer</i>	183
7.2.1.3.	<i>Hybrid Integer</i>	184
7.2.1.4.	<i>Comparative Analysis</i>	185
7.2.1.5.	<i>Runtime Performance</i>	187
7.2.2.	Heterozygous Chromosomes.....	188
7.2.2.1.	<i>Binary String</i>	188
7.2.2.2.	<i>Integer</i>	188
7.2.2.3.	<i>Hybrid Integer</i>	189
7.2.2.4.	<i>Comparative Analysis</i>	190
7.2.2.5.	<i>Runtime Performance</i>	195
7.2.3.	Caching.....	195
7.3.	Hanoi	196
7.3.1.	Genetic Representation.....	196
7.3.1.1.	<i>Binary String</i>	196
7.3.1.2.	<i>Integer</i>	197
7.3.1.3.	<i>Hybrid Integer</i>	198
7.3.1.4.	<i>Comparative Analysis</i>	199
7.3.1.5.	<i>Runtime performance</i>	201
7.3.2.	Caching.....	202
7.4.	Piedemonte San Germano	202
7.4.1.	Genetic Representation.....	202
7.4.1.1.	<i>Binary String</i>	202
7.4.1.2.	<i>Integer</i>	203

7.4.1.3. Hybrid Integer.....	204
7.4.1.4. Comparative Analysis.....	205
7.4.1.5. Runtime Performance.....	207
7.4.2. Heterozygous Chromosomes.....	207
7.4.2.1. Binary String.....	207
7.4.2.2. Integer.....	208
7.4.2.3. Hybrid Integer.....	209
7.4.2.4. Comparative Analysis.....	210
7.4.2.5. Runtime Performance.....	214
7.4.3. Caching.....	215
7.5. Conclusions.....	215
Chapter 8. Large Scale Optimization Problems.....	218
8.1. Introduction.....	218
8.2. “Real World” Network.....	218
8.2.1. Problem Formulation.....	218
8.2.2. Genetic Representation.....	220
8.2.2.1. Comparative Analysis.....	220
8.2.2.2. Runtime Performance.....	223
8.2.3. Caching.....	224
8.2.4. Distributed Performance.....	224
8.3. Stochastic Piedemonte San Germano.....	226
8.3.1. Problem Formulation.....	226
8.3.2. Non-Repeating Genetic Algorithm.....	227
8.3.3. Distributed Performance.....	230
8.4. Conclusions.....	231
Chapter 9. Conclusions.....	234
9.1. Further Research.....	238
Appendix A Network Infrastructure Modelling: OpenNet.....	244
A.1 Introduction.....	244
A.2 Implementation.....	246

A.3	Network Constituents	248
A.3.1	Elements.....	248
A.3.2	Element Lists and Stores.....	250
A.3.2.1	<i>Addition</i>	250
A.3.2.2	<i>Deletion</i>	251
A.3.2.3	<i>Searching</i>	251
A.3.2.4	<i>Other functionality</i>	252
A.3.3	Network.....	252
A.3.4	Node Elements	252
A.3.5	Link Elements	253
A.3.6	Element Type Registration	253
A.4	Hydraulic specialisation.....	256
A.5	Network analysis	259
A.5.1	Connectivity.....	259
A.5.2	Network Traversal.....	260
A.5.2.1	<i>Basic functions</i>	260
A.5.2.2	<i>Network Simplification</i>	263
A.6	Hydraulic Evaluation.....	264
A.6.1	Pressure Driven Demand.....	264
A.7	Extensions	265
A.7.1	Tracing.....	265
A.7.2	Generalized Attributes.....	265
A.8	Network model translation	266
A.8.1	Translation suite.....	266
A.8.2	Translator structure.....	267
A.8.3	User interface support	267
A.8.4	Difficulties.....	268
A.9	Linking hydraulic models to GIS applications	269
A.9.1	Pipe matching.....	271
A.9.1.1	<i>Import Hydraulic Network</i>	271

A.9.1.2 Import Asset Management Database	271
A.9.1.3 ID Matching	272
A.9.1.4 Geographic matching.....	272
A.9.1.5 Pipe grouping.....	273
A.9.1.6 Interactive matching	273
A.9.1.7 Results.....	275
A.10 Representing networks using XML	276
A.10.1 Elements	276
A.10.2 Entities	278
Appendix B OpenNet XML Representation.....	282
B.1 XML Schema.....	282
B.2 Example XML Network File	288
Bibliography	292
Papers presented by the candidate	292
Other Papers arising from this work	292
Published	292
In preparation.....	293
List of References	294

List of Tables

Table 3-1:	Comparison of conventional binary strings and Gray-coded binary strings for 4-bit values	60
Table 4-1:	Binary eXclusive OR (XOR) operation.....	75
Table 4-2:	Example dry run for decoding a Gray-coded binary string	76
Table 4-3:	Binary string implementations: relative performance.....	82
Table 4-4:	Comparison of cached/uncached performance for binary string representations	84
Table 4-5:	Comparison of Red-Black Binary Tree and Judy Tree cache requirements for New York Tunnels Problem (100,000 solutions)	97
Table 4-6:	Comparison of cached and best-case theoretical performance for the 200 job/20 agent GAP problem using the tiered Red-Black Binary Tree cache	98
Table 4-7:	Long term 500,000 evaluation comparison of cached and best-case run-times and evaluations	99
Table 4-8:	Runtime results for caching of small GAP 20 Agent/100 Job problem with variable mutation rates	101
Table 4-9:	Runtime results for caching of large GAP 20 Agent/200 Job problem with variable mutation rates	103
Table 5-1:	Hardware specifications of test environment computers.....	120
Table 5-2:	Baseline performance on Piedemonte San Germano simulation exercise. ...	122
Table 5-3:	Results obtained from running single threads on each of the computers and dual threads on the multiprocessor computers.....	124
Table 5-4:	Results utilizing one thread per processor (virtual or physical) plus one supplementary thread.....	125
Table 5-5:	Comparison of data transfer and performance for standard and devolved stochastic configurations (for the Piedemonte San Germano case study as before – assuming 50 stochastic samples).....	129
Table 6-1:	Hardware specifications of distributed test environment computers	135
Table 6-2:	New York Tunnels Node Characteristics.....	137
Table 6-3:	New York Tunnels Reservoir Characteristics.....	137
Table 6-4:	New York Tunnels Pipe Characteristics.....	138
Table 6-5:	New York Tunnels Pipe Duplication Options.....	138

Table 6-6:	New York Tunnels: Comparison with Literature Results	141
Table 6-7:	New York Tunnels Runtime Performance.....	143
Table 6-8:	New York Tunnels Heterozygous vs. Conventional Runtime Performance	148
Table 6-9:	Cache results: New York Tunnels	148
Table 6-10:	Theoretical maximum performance for distributed New York Tunnels problem	150
Table 6-11:	New York Tunnels distributed performance results	150
Table 6-12:	Hanoi Node Characteristics.....	153
Table 6-13:	Hanoi Reservoir Characteristics.....	153
Table 6-14:	Hanoi Pipe Characteristics	154
Table 6-15:	Hanoi Pipe Options	154
Table 6-16:	Hanoi Runtime Performance	158
Table 6-17:	Cache results: Hanoi.....	158
Table 6-18:	Theoretical maximum performance for distributed Hanoi problem.....	159
Table 6-19:	Hanoi distributed performance results	159
Table 6-20:	Comparison of optimal solutions to Hanoi problem	161
Table 6-21:	Piedemonte San Germano Node Characteristics	163
Table 6-22:	Piedemonte San Germano Reservoir Characteristics	163
Table 6-23:	Piedemonte San Germano Pipe Characteristics	164
Table 6-24:	Piedemonte San Germano Pipe Duplication Options	165
Table 6-25:	PSG Runtime Performance	169
Table 6-26:	PSG Heterozygous Runtime Performance vs. Conventional Performance.	174
Table 6-27:	Cache results: Piedemonte San Germano	174
Table 6-28:	Theoretical maximum performance for distributed Piedemonte San Germano problem	175
Table 6-29:	Piedemonte San Germano distributed performance results.....	177
Table 7-1:	<i>C</i> metrics for Multiple Objective New York Tunnels after 20 generations .	186
Table 7-2:	<i>C</i> metrics for Multiple Objective New York Tunnels after 100 generations	187
Table 7-3:	<i>C</i> metrics for Multiple Objective New York Tunnels after 1,000 generations	187
Table 7-4:	New York Tunnels Multiple Objective Runtime Performance.....	188

Table 7-5:	<i>C</i> metrics for Multiple Objective Heterozygous New York Tunnels after 20 generations	193
Table 7-6:	<i>C</i> metrics for Multiple Objective Heterozygous New York Tunnels after 100 generations	194
Table 7-7:	<i>C</i> metrics for Multiple Objective Heterozygous New York Tunnels after 1,000 generations	195
Table 7-8:	New York Tunnels Multiple Objective Heterozygous Runtime Performance vs. Conventional Performance	195
Table 7-9:	Cache results: Multiple Objective New York Tunnels.....	196
Table 7-10:	<i>C</i> metrics for Multiple Objective Hanoi after 20 generations	200
Table 7-11:	<i>C</i> metrics for Multiple Objective Hanoi after 100 generations	201
Table 7-12:	<i>C</i> metrics for Multiple Objective Hanoi after 1,000 generations	201
Table 7-13:	Hanoi Multiple Objective Runtime Performance	202
Table 7-14:	Cache results: Multiple Objective Hanoi	202
Table 7-15:	<i>C</i> metrics for Piedemonte San Germano after 20 generations.....	206
Table 7-16:	<i>C</i> metrics for Piedemonte San Germano after 100 generations.....	206
Table 7-17:	<i>C</i> metrics for Piedemonte San Germano after 1000 generations	207
Table 7-18:	Piedemonte San Germano Multiple Objective Runtime Performance	207
Table 7-19:	<i>C</i> metrics for Multiple Objective Heterozygous Piedemonte San Germano after 20 generations	213
Table 7-20:	<i>C</i> metrics for Multiple Objective Heterozygous Piedemonte San Germano after 100 generations	213
Table 7-21:	<i>C</i> metrics for Multiple Objective Heterozygous Piedemonte San Germano after 1,000 generations	214
Table 7-22:	PSG Multiple Objective Heterozygous Runtime Performance vs. Conventional Performance	214
Table 7-23:	Cache results: Multiple Objective Piedemonte San Germano.....	215
Table 8-1:	<i>S</i> metrics for Real World network after 100, 1,000 and 10,000 generations.....	222
Table 8-2:	<i>C</i> metrics for Real World network after 100 generations.....	223
Table 8-3:	<i>C</i> metrics for Real World network after 1,000 generations.....	223
Table 8-4:	<i>C</i> metrics for Real World network after 10,000 generations	223
Table 8-5:	Real World network Multiple Objective Runtime Performance.....	224
Table 8-6:	Cache results: Multiple Objective Real World problem	224

Table 8-7:	Theoretical maximum performance for distributed “Real World” problem	225
Table 8-8:	“Real World” distributed performance results.....	225
Table 8-9:	Theoretical maximum performance for distributed, stochastic Piedemonte San Germano problem.....	230
Table 8-10:	Stochastic Piedemonte San Germano distributed performance results	231
Table A-1:	Differences in network element representation between common hydraulic modelling packages	268

List of Figures

Figure 3-1:	Flowchart illustrating basic Genetic Algorithm operation.....	54
Figure 3-2:	Example of a chromosome using binary strings.....	59
Figure 3-3:	Example of a chromosome using integer values (same values as Figure 3-2)	59
Figure 3-4:	Conventional Binary String.....	60
Figure 3-5:	Outline class diagram for GA library implementation (Pascal version).....	65
Figure 3-6:	Final operational structure of generic GA implementation (single objective)	68
Figure 3-7:	GA methodology: final design (single objective).....	69
Figure 3-8:	GA methodology: final design (multiple objective).....	70
Figure 4-1:	C++ code fragment for encoding a binary string.....	74
Figure 4-2:	C++ code fragment for decoding binary string.....	74
Figure 4-3:	C++ code fragment for encoding a Gray-coded binary string.....	75
Figure 4-4:	Binary string prior to right shift.....	75
Figure 4-5:	Binary string following right shift.....	75
Figure 4-6:	C++ code fragment for decoding a Gray-coded binary string.....	76
Figure 4-7:	Parent gene a (value= 8,221).....	79
Figure 4-8:	Parent gene b (value= 392).....	79
Figure 4-9:	Expected child outputs from crossover.....	79
Figure 4-10:	Crossover mask c (value= 255).....	80
Figure 4-11:	C++ code to generate mask for crossover.....	80
Figure 4-12:	Masked parent d – least significant byte (value= 29).....	80
Figure 4-13:	Masked parent e - least significant byte (value= 136).....	80
Figure 4-14:	Masked parent f - most significant byte (value= 8,192).....	80
Figure 4-15:	Masked parent g - most significant byte (value= 256).....	81
Figure 4-16:	Child h (value= 8,328).....	81
Figure 4-17:	Child i (value= 285).....	81
Figure 4-18:	Flowchart illustrating the role of caching in a simple GA.....	86
Figure 4-19:	Traditional binary tree representation.....	87
Figure 4-20:	Unbalanced binary tree.....	88

Figure 4-21:	Example Red-Black binary tree.....	89
Figure 4-22:	Pseudo-code for cache search logic	90
Figure 4-23:	Tiered Cache.....	91
Figure 4-24:	Two-way Digital Tree (trie).....	92
Figure 4-25:	Three-way Digital Tree (trie)	92
Figure 4-26:	Example Binary Tree representation	94
Figure 4-27:	Example Judy Tree representation demonstrating implicit compression of search key.....	95
Figure 4-28:	Example New York Tunnels chromosome.....	96
Figure 4-29:	Digital/Judy tree implementation for New York Tunnels chromosome	96
Figure 4-30:	Algorithmic performance (median) for small GAP problem (20 Agent/100 Job) with variable mutation rates.....	100
Figure 4-31:	Comparative Runtimes for small GAP problem (20 Agent/100 Job) with variable mutation rates and four caching strategies	101
Figure 4-32:	Algorithmic performance (median) for large GAP problem (20 Agent/200 Job) with variable mutation rates.....	102
Figure 4-33:	Comparative Runtimes for large GAP problem (20 Agent/200 Job) with variable mutation rates and four caching strategies	103
Figure 4-34:	Outline flowchart for non-repeating GA.....	105
Figure 4-35:	C++ structure for recording gene mutation trend score data	107
Figure 4-36:	C++ code for trend scoring for differential mutation	108
Figure 4-37:	C++ code for mutation operator	108
Figure 4-38:	C++ code for differential mutation operator	109
Figure 5-1:	Typical PC network configuration for deploying deEPANET	116
Figure 5-2:	Topology of Piedemonte San Germano Case Study Network	121
Figure 5-3:	Logical Structure of Stochastic Optimization Software (after Kapelan, 2005)	122
Figure 5-4:	Baseline performance on Piedemonte San Germano simulation exercise. ..	123
Figure 5-5:	Results utilizing one thread per processor (virtual or physical) plus one supplementary thread.	126
Figure 5-6:	Extended test network for deEPANET simulations.....	128
Figure 6-1:	Example result graph	132
Figure 6-2:	Mutation performance comparison - Generalized Assignment Problem.....	134

Figure 6-3:	New York Tunnels Topology.....	136
Figure 6-4:	Algorithmic Performance: New York Tunnels - Binary String.....	139
Figure 6-5:	Algorithmic Performance: New York Tunnels - Gray Binary String.....	140
Figure 6-6:	Algorithmic Performance: New York Tunnels – Integer.....	140
Figure 6-7:	Algorithmic Performance: New York Tunnels - Hybrid Binary String.....	141
Figure 6-8:	Algorithmic Performance: New York Tunnels - Combined Best.....	142
Figure 6-9:	Algorithmic Performance: New York Tunnels - Combined Upper/Lower Quartiles.....	142
Figure 6-10:	Algorithmic Performance: New York Tunnels - Heterozygous Binary String	144
Figure 6-11:	Algorithmic Performance: New York Tunnels - Heterozygous Binary String results overlain with conventional results.....	144
Figure 6-12:	Algorithmic Performance: New York Tunnels - Heterozygous Integer	145
Figure 6-13:	Algorithmic Performance: New York Tunnels - Heterozygous Integer results overlain with conventional results	145
Figure 6-14:	Algorithmic Performance: New York Tunnels - Heterozygous Hybrid Binary String	146
Figure 6-15:	Algorithmic Performance: New York Tunnels - Heterozygous Hybrid Integer results overlain with conventional results.....	146
Figure 6-16:	Algorithmic Performance: New York Tunnels – Combined Heterozygous Best.....	147
Figure 6-17:	Algorithmic Performance: New York Tunnels - Combined Heterozygous Upper/Lower Quartiles.....	147
Figure 6-18:	Mutation performance comparison - New York Tunnels problem.....	149
Figure 6-19:	Hanoi Network Topology.....	152
Figure 6-20:	Algorithmic Performance: Hanoi - Binary String	155
Figure 6-21:	Algorithmic Performance: Hanoi – Integer	156
Figure 6-22:	Algorithmic Performance: Hanoi - Hybrid Integer	156
Figure 6-23:	Algorithmic Performance: Hanoi - Combined Best.....	157
Figure 6-24:	Algorithmic Performance: Hanoi - Combined Upper/Lower Quartiles	157
Figure 6-25:	Mutation performance comparison - Hanoi.....	158
Figure 6-26:	Piedemonte San Germano Network Topology	162
Figure 6-27:	Algorithmic Performance: PSG - Binary String.....	166

Figure 6-28:	Algorithmic Performance: PSG – Integer	167
Figure 6-29:	Algorithmic Performance: PSG - Hybrid Integer	167
Figure 6-30:	Algorithmic Performance: PSG - Combined Best	168
Figure 6-31:	Algorithmic Performance: PSG - Combined Upper/Lower Quartiles.....	168
Figure 6-32:	Algorithmic Performance: PSG – Heterozygous Binary String	169
Figure 6-33:	Algorithmic Performance: PSG- Heterozygous Binary String results overlain with conventional results	170
Figure 6-34:	Algorithmic Performance: PSG – Heterozygous Integer	170
Figure 6-35:	Algorithmic Performance: PSG- Heterozygous Integer results overlain with conventional results	171
Figure 6-36:	Algorithmic Performance: PSG – Heterozygous Hybrid Integer	171
Figure 6-37:	Algorithmic Performance: PSG- Heterozygous Hybrid Integer results overlain with conventional results.....	172
Figure 6-38:	Algorithmic Performance: PSG – Heterozygous Combined Best.....	173
Figure 6-39:	Algorithmic Performance: PSG - Combined Upper/Lower Quartiles.....	173
Figure 6-40:	Mutation performance comparison - Piedemonte San Germano	175
Figure 7-1:	Multiple Objective Pareto-Optimal Front	180
Figure 7-2:	New York Tunnels – Multiple Objective Binary String Results.....	183
Figure 7-3:	New York Tunnels – Multiple Objective Integer Results	184
Figure 7-4:	New York Tunnels – Multiple Objective Hybrid Integer Results	185
Figure 7-5:	Box plots of S metric for Multiple Objective New York Tunnels after 20, 100 & 1,000 generations	185
Figure 7-6:	New York Tunnels – Multiple Objective Heterozygous Binary String Results	188
Figure 7-7:	New York Tunnels – Multiple Objective Heterozygous Integer Results	189
Figure 7-8:	New York Tunnels – Multiple Objective Heterozygous Hybrid Integer Results.....	190
Figure 7-9:	Graphical Comparison of Multiple Objective New York Tunnels results...	191
Figure 7-10:	Box plots of S metric for Multiple Objective Heterozygous New York Tunnels after 20, 100 & 1,000 generations.....	192
Figure 7-11:	Box plots of S metric for Multiple Objective New York Tunnels for Normal and Heterozygous New York Tunnels after 20, 100 & 1,000 generations	192
Figure 7-12:	Hanoi – Multiple Objective Binary String Results	197

Figure 7-13:	Hanoi – Multiple Objective Integer Results	198
Figure 7-14:	Hanoi – Multiple Objective Hybrid Integer Results	198
Figure 7-15:	Graphical Comparison of Multiple Objective Hanoi results	199
Figure 7-16:	Box plots of S metric for Multiple Objective Hanoi after 20, 100 and 1,000 generations	200
Figure 7-17:	Piedemonte San Germano– Multiple Objective Binary String Results	203
Figure 7-18:	Piedemonte San Germano– Multiple Objective Integer Results	204
Figure 7-19:	Piedemonte San Germano– Multiple Objective Hybrid Integer Results	204
Figure 7-20:	Box plots of S metric for Multiple Objective Piedemonte San Germano after 20, 100 and 1,000 generations	205
Figure 7-21:	Piedemonte San Germano– Multiple Objective Heterozygous Binary String Results	208
Figure 7-22:	Piedemonte San Germano– Multiple Objective Heterozygous Integer Results	209
Figure 7-23:	Piedemonte San Germano– Multiple Objective Heterozygous Hybrid Integer Results	209
Figure 7-24:	Graphical Comparison of Multiple Objective Piedemonte San Germano results	210
Figure 7-25:	Box plots of S metric for Multiple Objective Heterozygous Piedemonte San Germano after 20, 100 & 1,000 generations.....	211
Figure 7-26:	Box plots of S metric for Multiple Objective Normal and Heterozygous Piedemonte San Germano after 20, 100 & 1,000 generations	212
Figure 8-1:	"Real World" network topology.....	219
Figure 8-2:	Best Pareto fronts for "Real World" problem after 100 generations	220
Figure 8-3:	Best Pareto fronts for "Real World" problem after 1,000 generations.....	221
Figure 8-4:	Best Pareto fronts for "Real World" problem after 10,000 generations.....	222
Figure 8-5:	Non-Repeating GA performance comparison.....	229
Figure A-1:	New York Tunnels-specific version of GAnet with OpenNet visualization component	245
Figure A-2:	Constituents of a network representation	246
Figure A-3:	OpenNet class hierarchy (partial).....	247
Figure A-4:	OpenNet pipe properties dialog box.....	254
Figure A-5:	High level class hierarchy of OpenNet implementation.....	255
Figure A-6:	Typical 24 hour domestic demand curve.....	257

Figure A-7: Recursive network traversal..... 262

Figure A-8: Recursive network traversal (continued)..... 263

Figure A-9: Network schematic simplification with OpenNet..... 263

Figure A-10: UML Class Hierarchy for generalized attributes..... 265

Figure A-11: Translation options available through OpenNet..... 267

Figure A-12: Translator progress window showing a SynerGEE model being imported
into OpenNet..... 268

Figure A-13: Pipe matching application..... 274

Glossary

Definitions

- Allele** The *alleles* of a gene are the set of values that this gene can take. The most straightforward example taken from biology is that of the gene that determines eye colour. The alleles of this gene are the different colours (Brown, Blue...etc...)
- Application Programming Interface** A specification for the public interface to a software library to be used by developers.
- Chromosome** Many workers in the field of evolutionary algorithms use the term chromosome instead of *organism* above. As will be shown, it is convenient to maintain a distinction between the two (as well as maintaining the biological analogue) and to preserve the *chromosome* moniker to describe a group of *genes* that are related in some fashion. Thus, a *chromosome* is some sub-division of an *organism's genome*.
- Common Object Model** A technology developed by Microsoft for the implementation of a componentized software architecture featuring a standardized API for the introspection of methods and members. Microsoft Windows specific. Largely supplanted by the .NET technology but still underpinning many Microsoft Products (e.g. Office).
- Distributed Common Object Model.** As *COM* but with additional functionality to allow the instantiation of objects remotely, across a network.
- Dynamic Link Library** A mechanism for dynamically linking software functions into an application. Promotes componentization through code reuse – applications can be composed from smaller building blocks. Unlike COM, however, DLLs do not have a standard mechanism for implementing object classes – which has

resulted in compiler vendors implementing their own, incompatible techniques for doing so.

- Elitism Applied to generational GAs, elitism allows the most fit individuals from a source population to transfer directly to the destination population without undergoing recombination – ensuring that they are preserved from generation to generation.
- Fitness The measure by which individual organisms are compared to each other to judge their relative suitability for the optimization problem at hand. For single objective algorithms this is a single value, often combined with a *penalty function* to penalise constraint violation.
- Gene Genes are the fundamental unit of genetic algorithms. Each gene represents a specific attribute that is encoded within the genome at a specific location known as a *locus*. This attribute normally represents a decision variable to be considered in the optimization but can also convey other information specific to an *organism*.
- Generational GA A type of Genetic Algorithm used for both single and multiple objective optimization. Following selection and recombination, child organisms are inserted into a *new* population rather than replaced into the existing population as with a Steady-State GA. The selection and recombination process continues until the new population reaches the nominal size of the previous generation. The new population then forms the basis for selection for the next generation.
- Genome The total genetic representation of an organism is described as its genome. In mammalian biology, a genome is ordinarily sub-divided into chromosomes.
- Genotype The representation of the information contained within the genome in its native form. For example, a decision variable

	containing the value 5 may represent a pipe diameter of 80mm. The genotypic value of the decision variable is 5.
Locus	The locus of a <i>gene</i> is the position it occurs in a <i>chromosome</i> . In Genetic Algorithms, the locus is usually fixed for a given gene. However, in certain types of GA with variable-length or <i>heterozygous chromosomes</i> the locus can vary.
Network Calibration	An optimization problem to calibrate a Water Distribution System hydraulic model to match observed field data – commonly by varying pipe friction factors or diameters.
Network Design	An optimization problem to layout new pipes for a Water Distribution System.
Network Rehabilitation/Reinforcement	An optimization problem that determines intervention strategies in a WDS for rehabilitating existing pipes or reinforcing the network through pipe duplication.
Organism	This is the representation of an individual in the <i>population</i> . It is a term not normally associated with genetic algorithms and is a by-product of the underlying object-oriented library described here. Conventionally, the term “chromosome” has been the preferred term for an individual in GAs. However, because this use of chromosome is significantly at variance with the biological analogue – and the necessity for the object-oriented library to have an appropriate naming strategy for classes – it was decided to break with convention and to use “ <i>organism</i> ” as the fundamental unit of a <i>population</i> instead. Because each <i>organism</i> always has exactly one <i>genome</i> and that <i>genome</i> can only belong to a single <i>organism</i> they can be considered concomitant.
Penalty Function	A function commonly used in single objective optimization to represent constraint violation by an individual organism – normally used in combination with a <i>fitness</i> value to give an overall relative measure of fitness for a solution.

Phenotype	The representation of the information contained within the genome in its applied form. For example, a decision variable containing the value 5 may represent a pipe diameter of 80mm. The genotypic value of the decision variable is 80 when translated into the domain of the solution.
Population	A collection of organisms – the pool of genetic material operated on by a Genetic Algorithm.
Recombination	The derivation of child solutions from their parents – ordinarily as a result of crossover and mutation.
Steady-state GA	A type of single-objective Genetic Algorithm in which a pair of solutions are selected from a population, recombined to form two children and those children inserted, according to some rule, back into the original population. c.f. Generational GA.

List of Abbreviations

ACS	Ant Colony Simulation
ADSL	Asymmetric Digital Subscriber Line
AMS	Asset Management System
ANSI	American National Standards Institute
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
CA	Cellular Automata
COM	Common Object Model.
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CWS	Centre for Water Systems, University of Exeter
DCOM	Distributed Common Object Model
DLL	Dynamic Link Library
DMA	District Metered Area
DSS	Decision Support System
DTD	Document Type Definition (XML)
EPS	Extended Period Simulation
FLV	Float Valve
GA	Genetic Algorithm
GAP	Generalized Assignment Problem
GIS	Geographic Information System
IP	Internet Protocol
ISO	International Standards Organisation
LAN	Local Area Network
LP	Linear Programming

LSB	Least Significant Bit (binary number)
MGA	Messy Genetic Algorithm
MOGA	Multiple Objective Genetic Algorithm
MSB	Most Significant Bit (binary number)
MTV	Motorised Throttle Valve
NLP	Non-Linear Programming
NRGA	Non-Repeating Genetic Algorithm
NRV	Non-Return Valve
NSGA	Non-dominated Sorted Genetic Algorithm
NYT	New York Tunnels (benchmark problem)
OLE	Object Linking & Embedding
OOTEN	Object-Oriented Toolkit for EPANET
OSGB	Ordnance Survey of Great Britain
PBV	Pressure Break Valve
PDD	Pressure-Driven Demand
PDF	Probability Density Function
PRV	Pressure Reducing Valve
PSG	Piedmonte San Germano (benchmark problem)
PSO	Particle Swarm Optimization
PSV	Pressure Sustaining Valve
RCV	Remote Control Valve
RDBMS	Relational Database Management Software
rNSGA-II	Robust NSGA-II
SA	Simulated Annealing
SDSS	Spatial Decision Support System
SFLA	Shuffled Frog Leaping Algorithm

SGML	Standard Generalized Markup
SMGA	Structured, Messy Genetic Algorithm
SMP	Symmetric Multi-Processing
SOGA	Single Objective Genetic Algorithm
STL	Standard Template Library
SWMM	Storm Water Management Model
TCP	Transmission Control Protocol
THV	Throttle Valve
UDP	User Datagram Protocol
UML	Unified Modelling Language
WAN	Wide Area Network
WAP	Wireless Access Point
WDS	Water Distribution Systems
XML	eXtensible Markup Language
XOR	eXclusive OR (<i>logical operation</i>)

