

Optimisation of Short Term Conflict Alert Safety Related Systems

William Reckhouse

School of Engineering, Computing and Mathematics,

University of Exeter

October 2010

Submitted by William James Reckhouse, to the University of Exeter as a thesis for the degree of Doctor of Philosophy in Computer Science, October 2010.

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.

.....

Abstract

Short Term Conflict Alert (STCA) is an automated warning system designed to alert air traffic controllers to possible loss of separation between aircraft. STCA systems are complex, with many parameters that must be adjusted to achieve best performance. Current procedure is to manually ‘tune’ the governing parameters in order to finely balance the trade-off between wanted alerts and nuisance alerts.

We present an incremental approach to automatically optimising STCA systems, using a simple evolutionary algorithm. By dividing the parameter space into regional subsets, we investigate methods of reducing the number of evaluations required to generate the Pareto optimal Receiver Operating Characteristic (ROC) curve. Multi-archive techniques are devised and are shown to cut the necessary number of iterations by half. A method of estimating the fitness of recombined regional parameter subsets without actual evaluation on the STCA system is presented, however, convergence is shown to be severely stunted when relatively weak sources of noise are present.

We describe a method of aggressively perturbing parameters outside of their known ‘safe’ ranges when complex inhibitory interactions are present that prevent an exhaustive search of permitted values. The scheme prevents the optimiser from repeating ‘mistakes’ and unnecessarily wasting evaluations. Results show that a more complete picture of the Pareto-optimal ROC curve may be obtained without increasing the number of necessary iterations.

Efficacy of the new methods is discussed, with suggestions for improving efficiency. Sources of parameter interdependence and noise are explored and where possible mitigating techniques and procedures suggested. Classifier performance on training and test data is investigated and potential solutions for reducing overfitting are evaluated on a toy problem. We comment on potential uses of the ROC in characterising STCA performance, for comparison to other systems and airspaces.

Many industrial systems are structured in a similar way to STCA, we hope that techniques presented will be applicable to other highly parametrised, expensive problem domains.

Acknowledgements

This work was supported by a Knowledge Transfer Partnership awarded to NATS and the University of Exeter.

Special thanks to:

Richard Everson* and Jonathan Fieldsend* for all their key input throughout the project and prior supporting work.

Trevor Arnold[†]

Richard Hayward[†]

Keith Slater[†]

David Bush[†]

and Rod Bacon[†] & Lawrence Gillet[†] of the ‘ARDAT team’ for their support with OpenVMS.

My partner Jennifer Vooght for all her support and patience throughout.

*Computer Science, College of Engineering, Mathematics and Physical Sciences, University of Exeter.

[†]Operational Analysis Department, NATS.

Contents

1	Introduction	9
1.1	Main contributions	11
1.2	Publications	12
2	Air traffic control background	13
2.1	Airspace	13
2.2	Radar	14
2.3	Short Term Conflict Alert (STCA)	15
2.3.1	STCA technical overview	16
2.3.2	Sectorisation	17
2.3.3	Manual optimisation of STCA	19
2.3.4	Enhanced Short Term Conflict Alert (ESTCA)	20
2.3.5	STCA parameters	20
2.3.6	The offline STCA environment	22
2.4	Automated optimisation of multiple objectives	23
2.4.1	ROC analysis	24
2.4.2	Automated optimisation of STCA	26
2.4.3	Application of ROC analysis and Pareto Optimality	28
2.4.4	Tabu search	30
2.4.5	Multi-objective evolutionary algorithms	31
2.5	Chapter summary	44
3	The single-archive optimiser	45
3.1	The single-archive optimiser	45
3.1.1	Initialisation of the archive	47
3.1.2	Parent selection	48
3.1.3	Perturbation	50
3.1.4	Results of the single-archive optimiser	52
3.1.5	Optimisation speed	54
3.2	Summary	55

4	A multi-archive approach to optimisation	56
4.1	The multi-archive optimiser	56
4.1.1	‘Global’ Uniselect selection	61
4.1.2	Results of the multi-archive optimiser	65
4.1.3	Summary	67
4.2	The estimated-archive	68
4.2.1	Results of the estimated-archive	72
4.2.2	Summary	75
4.3	Discussion of efficacy	76
4.3.1	Cross-regional aircraft pairs & parameter independence	76
4.3.2	Dataset regional labelling discrepancies	77
4.3.3	Correcting categoriser region-labelling discrepancies	78
4.3.4	The region 0’s discrepancy	87
4.3.5	Discrepancies summary	87
4.4	Estimated-archive revisited	89
4.4.1	Results of the modified estimated-archive	90
4.4.2	Summary	93
4.5	Summary	93
5	Aggressive optimisation	94
5.1	Aggressive optimisation	94
5.1.1	Results of the aggressive optimiser	97
5.2	Summary	99
6	Single-parent optimisation and non-regional parameters	103
6.1	Single-parent optimisation	103
6.2	The single-parent, single-archive optimiser	104
6.2.1	Results of the single-parent, single-archive optimiser	105
6.2.2	Summary	107
6.3	The single-parent, multi-archive optimiser	107
6.3.1	Results of the single-parent, multi-archive optimiser	110
6.3.2	Summary	112
6.4	Discussion of efficacy 2	113
6.4.1	Non-regional parameter dependencies	113
6.4.2	Single-parent, multi-archive, non-regionals non-optimisable	113
6.4.3	Multi-parent, multi-archive, non-regionals non-optimisable	115
6.4.4	Estimated-archive, non-regionals non-optimisable	117
6.4.5	Analysis of non-regional parameter values	118
6.4.6	Summary	125
6.5	Chapter summary	126

7	Classifier over-fitting	127
7.1	STCA classifier variance	127
7.2	Experimentation on a ‘toy’ problem	128
7.2.1	Single-archive trial (base-case)	134
7.2.2	Two archive cross-validation with swap	135
7.2.3	Two archive cross-validation no swap	138
7.2.4	Three archive cross-validation no swap	140
7.2.5	Bootstrap evaluation	142
7.2.6	Randomise evaluation	147
7.2.7	Probabilistic archive	149
7.2.8	Summary of over-fitting mitigation methods	153
7.3	Reducing STCA classifier over-fitting	154
8	Conclusions	157
8.1	Applications	157
8.1.1	Comparison of STCA systems: Optimisation of ‘Enhanced STCA’ . . .	157
8.1.2	Visualisation	161
8.1.3	Airspace review	161
8.1.4	Summary	162
8.2	Conclusions	164
8.3	Suggestions for future work	166
	Appendices	169
A		170
A.1	Example STCA parameter configuration file	170
B		171
B.1	Optimiser data flow diagram	171
C		172
C.1	Wrapping the STCA system	172
C.1.1	STCA architecture	172
C.1.2	Optimiser’s server-wrapper architecture	173
D		176
D.1	AR DAT CPU Usage Summary	176
D.1.1	Optimiser configuration	176
D.1.2	Monitor setup	176
D.1.3	Results	177
D.1.4	Summary	182

E	183
E.1 Stills from optimiser mode comparison video	183
References	186

Common abbreviations

ANSP	Air Navigation Service Provider
ATC	Air Traffic Control
ATCO	Air Traffic Control Officer
CAA	Civil Aviation Authority
CAMPAP	Conflict Alert Management Performance Analysis Package
COP	Current Operating Point
CPA	Closest Point of Approach
CPU	Central Processing Unit
ESTCA	Enhanced STCA
FP	False Positive
LACC	London Area Control Centre
LTCC	London Terminal Control Centre
MACC	Manchester Area Control Centre
MOEA	Multi-Objective Evolutionary Algorithm
OS	Operating System
PSR	Primary Surveillance Radar
ROC	Receiver Operating Characteristic
ScOACC	Scottish Area Control Centre
SFL	Selected Flight Level
SSR	Secondary Surveillance Radar
STCA	Short Term Conflict Alert
TP	True Positive
XML	Extensible Markup Language

Chapter 1

Introduction

NATS is the UK's main Air Navigation Service Provider (ANSP), responsible for handling all 'en route' flights through UK airspace and providing Air Traffic Control (ATC) services at 15 of the largest UK airports. All 'en route' business is regulated by the Civil Aviation Authority (CAA), while external contracts for engineering, technical and training services related to Air Traffic Control (ATC) are awarded after competitive bidding.

NATS' business is focused on ensuring the safety of aircraft. The Short Term Conflict Alert (STCA) system is used at NATS to alert air traffic controllers to potential conflicts between aircraft which, in a worst-case scenario, might otherwise result in a dangerous loss of separation between the aircraft. STCA is essentially a piece of automated software that processes aircraft tracks derived from ground-based radar and raises a visual alert to controllers, highlighting conflict pairs via the radar display. Controllers can then take appropriate action to resolve the conflict.

The recent European move towards standardisation of STCA-type systems has meant that by the end of 2008 all European ANSPs were expected to implement a system which at least conforms to the minimum requirements laid out in the Eurocontrol specification [32][31]. STCA systems are essentially classifiers that classify aircraft pairs into 'operationally relevant alerts' or true positives (TP) and 'nuisance alerts' or false positives (FP) - alerts that unnecessarily attract the controller's attention. The Eurocontrol mandate stipulates that an STCA system shall provide for operationally relevant conflicts, while keeping false or nuisance alerts to an effective minimum. Central to the safety case for STCA are two criteria:

1. "the proportion of conflicts detected by the controller in time for controller resolution will be enhanced by the use of STCA" [32];
2. "any negative effects on safety shall be small compared with the safety benefit and reduced as far as reasonably practical" [32].

The challenge for analysts and maintenance personnel is in making informed decisions about the system's likely performance and selecting an appropriate trade-off between 'operationally relevant' alerts, and those which are deemed nuisance or false. STCA systems are generally complex and tuning them involves adjustment of a great many (typically over

1000) operational parameters. Current procedure is to manually ‘tune’ the parameters in order to optimise the operating point, however this is a lengthy and involved process.

This thesis describes how automation of the optimisation process, using a multi-objective evolutionary algorithm, can produce a good approximation to the set of parametrisations yielding best possible trade-offs between true and false positive rates. We describe methods of reducing the number of necessary candidate evaluations in order to speed up optimisation and explore potential uses for the optimal set of trade-offs once obtained. We explore just how well optimised parametrisations generalise to unseen data, investigating how variance between training and test dataset true/false positive rates might be reduced using a toy neural-net problem.

Chapter 2 provides a background to ATC and describes the techniques used at NATS in the implementation of an STCA system. Optimisation is cast as a multi-objective problem suitable for automated exploration using an evolutionary algorithm.

In chapter 3 the description of a basic optimiser and initial results are documented; the architecture presented here forms the basis of future enhancements discussed in subsequent chapters. Although the algorithm used is capable of automated optimisation we find the process is slow, inhibited by the vast STCA parameter space to be explored. We seek to address limitations by dividing STCA parameters into several regional subsets and developing a novel ‘multi-archived’ algorithm to tune these subsets in parallel (chapter 4, section 4.1); results from this modified algorithm are shown to be a marked improvement, requiring far fewer STCA evaluations.

In an attempt to further reduce optimisation time, the concept of multiple archives can be extended by generating a set of ‘estimated’ parametrisations that need not be evaluated on the STCA system (chapter 4, section 4.2). We show how estimation of a parametrisation’s true and false positive rate can be made by recombining previously evaluated parameter subsets.

Limitations of the multi-archive and estimated-archive approaches are discussed in chapter 4, section 4.3, where we describe how dataset region labelling discrepancies and airspace construction can have an adverse effect on parameter tuning, introducing noise. We suggest how current dataset compilation procedure could be amended to minimise the problem.

Chapter 5 details how the algorithms can be extended to ‘aggressively’ tune STCA parameters outside of their known ‘safe’ ranges. Where previous trials have restricted perturbation to ranges defined by analysing parametrisations previously used by NATS as the operating point, the aggressive implementation allows safe exploration of previously unexplored parameter ranges. The algorithm is designed to record any values that cause STCA to become unstable, marking such boundaries as unsafe for future use. Results show that use of this algorithm can provide a more complete picture of the trade-off between true and false positives at higher rates and enhance potential conflict detection rates.

A modified selection method is investigated in chapter 6. In this alternative approach the optimisation ‘state’ is represented by a single parent parametrisation that is repeatedly perturbed, rather than an archive of parents from which members are selected at random. We apply this method to both single and multi-archive optimisation modes, comparing the

results to earlier trials. Further limitations are discussed in chapter 6, section 6.4, where we investigate the effect of parameter dependencies, in particular those concerning the ‘non-regional’ parameters, on optimisation efficacy.

In chapter 7 we describe how over-fitting in classifiers can cause a large variance in the performance on unseen test data. We perform experiments using a ‘toy’ neural-net problem in order to devise and assess techniques designed to alleviate the issue and discuss how such techniques might be applied to reducing overfitting during optimisation of STCA.

Finally we discuss potential applications for results of the automated STCA optimisation process in visualisation, airspace review and comparison of STCA-type systems from different vendors (chapter 8).

1.1 Main contributions

The main contributions of this work are:

- Multi-objective optimisation of STCA systems.
We illustrate how optimisation of STCA systems can be automated for multiple objectives using an evolutionary algorithm (chapter 2 section 2.4.2 introduces the concept, chapter 3 section 3.1 describes an initial implementation and results).
- A multi-archived approach to tuning Receiver Operating Characteristic (ROC) curves.
We show how the regional structure of STCA parametrisations can be exploited by maintaining multiple archives to optimise parameter subsets simultaneously (chapter 4 section 4.1).
- A method of estimating parametrisation results.
We describe how the multi-archived approach to optimising parameter subsets can be extended to estimate the fitness of parametrisations without evaluation on the STCA system (chapter 4 section 4.2).
- An algorithm for aggressive perturbation in unknown parameter ranges.
We demonstrate a method of safely exploring unknown STCA parameter ranges in an intelligent manner (chapter 5 section 5.1).
- Preliminary investigation into reducing over-fitting in STCA classifiers.
We present an initial comparison of techniques for reducing overtraining in classifiers (chapter 7 section 7.1).

1.2 Publications

Parts of the work in this thesis have been reported in the following publications:

W.J. Reckhouse, R.M. Everson, J.E. Fieldsend, D. Bush, T. Arnold, R. Hayward, and K. Slater. Assessment and optimisation of STCA performance: Using the Pareto optimal receiver operating characteristic. In *Proceedings of Eurocontrol Annual Safety R&D Seminar*, Southampton, UK, 22–24 October 2008.

W.J. Reckhouse, J.E. Fieldsend, and R.M. Everson. Variable interactions and exploring parameter space in an expensive optimisation problem: optimising short term conflict alert. *IEEE CEC 2010*, Barcelona, July 2010.

Chapter 2

Air traffic control background

To describe key issues involved in implementation of an STCA-type system, certain background subject matter must be covered. In this chapter an introduction to the pertinent topics will be given, introducing the STCA tool and the technologies behind it.

2.1 Airspace

It is NATS' responsibility to ensure the safe separation of aircraft in UK airspace, to help simplify the task airspace is divided into several different categories each designed to facilitate manageable aircraft patterns.

Broadly speaking there are two main types of airspace; controlled and uncontrolled or 'free' airspace. Controlled and uncontrolled airspace in the UK is shown in figure 2.1, where controlled airspace has been marked as coloured blocks, uncoloured regions are uncontrolled airspace. In controlled airspace, it is the Air Traffic Control Officer's (ATCO's) responsibility to supervise separation of aircraft at all times. ATC may provide advisory traffic information services in uncontrolled airspace, but under normal circumstances, they have no authority to intervene with aircraft.

For air traffic control purposes the entirety of UK airspace is divided up into two Flight Information Regions (FIRs); Scottish FIR and London FIR (see figure 2.1). These are further broken down into different 'sectors', each with a team of ATCOs responsible for handling traffic from one of four Control Centres: Scottish and Oceanic Area Control Centre (ScOACC) based at Prestwick; Manchester Area Control Centre (MACC) at Manchester Airport; London Terminal Control Centre (LTCC) and London Area Control Centre (LACC) now sharing the same site at Swanwick (as of November 2007 when LTCC operations ceased at West Drayton).

Controlled airspace consists of 'airways', 'terminal control zones' and 'airport control zones' (see figure 2.1), each of which can be divided into sectors. Airways act as the main aviation routes between terminal control areas (established where airways meet in the vicinity of airports). Airport control zones extend from the ground to a specified height, and control of flights is passed from en-route ATCOs to airport ATC personnel as they enter the zone. The division of airspace is designed to force aircraft into expected behaviour patterns thus

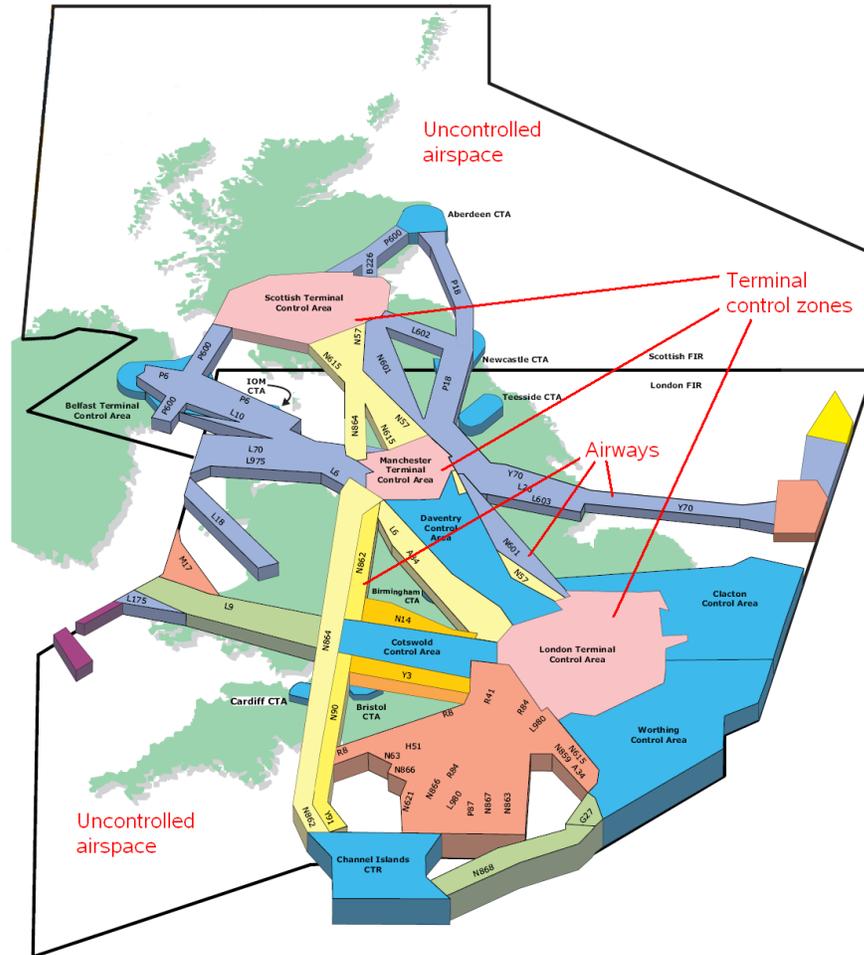


Figure 2.1: UK airways and terminal control areas. Source: NATS, 2008.

simplifying the task of ATC. During quieter periods two or more sectors may be ‘bandboxed’ together allowing one controller to oversee several sectors at once.

2.2 Radar

The progress of aircraft in controlled airspace is monitored by ground based radar. Two types of radar are involved in the process of acquiring aircraft information. Primary Surveillance Radar (PSR) operates independently of aircraft. That is to say, it does not require specialist apparatus on board the aircraft. PSR works by transmitting a pulse of energy, part of the energy is reflected back from any object in its path; returned energy is detected and can be used to measure range and bearing.

Secondary Surveillance Radar (SSR) is more advanced than PSR in that it requests information from the aircraft on its identity (Mode-A data) and altitude (Mode-C data). The altitude information is particularly important because, unlike range and bearing, height measurement from ground based radar is relatively inaccurate. However, unlike PSR, SSR is not passive and it requires the aircraft to be fitted with a ‘transponder’ to transmit the

requested information. SSR transmits a pulse of energy that triggers the transponder in the aircraft to broadcast data back to the radar. All commercial aircraft in UK airspace are required to have a Mode-A+C transponder fitted [34].

In addition to Mode-A and C data, a new packet standard called ‘Mode-S’ is also being increasingly used, however not all aircraft have it equipped. Mode-S data is a packet standard for both uplink (to aircraft) and downlink (from aircraft) information [33]. It can be used to obtain additional data on, for instance, an aircraft’s Selected Flight Level (SFL), the level to which an aircraft is currently climbing or descending. SFL information is not used in the STCA ISS4 implementation, however it may be used to help reduce nuisance alerts in the newer Enhanced STCA (ESTCA) that is in development at NATS [66].

2.3 Short Term Conflict Alert (STCA)

It is the ATCO’s job to ensure that separation standards are adhered to: in the UK this means that aircraft should ordinarily be separated by 5 nautical miles horizontally and 1000 feet vertically. However, the required horizontal separation may vary dependant on location and radar cover; around airports, for instance, the horizontal separation can be reduced to 3 nautical miles. This means that there is less room for error when manoeuvring aircraft, as conflicts must be spotted and resolved in a shorter time frame.

The Short Term Conflict Alert (STCA) system is used to alert air traffic controllers to potential conflicts between aircraft which, in a worst-case scenario, might otherwise result in a loss of separation. STCA is essentially a piece of automated software that processes aircraft tracks and raises a visual alert to controllers, highlighting conflict pairs via the radar display. Controllers can then take appropriate action. STCA is considered a safety net, *not* a safety critical system; it is designed to improve safety, alerting to any potential conflicts that might arise in the next ≈ 2 minutes. However, controllers do not have to act on an alert, some conflicts raised by the STCA system may in fact be ignored as erroneous or nuisance alerts. This is why the distinction between safety-critical and safety-net is made; STCA improves safety, however the safe navigation of aircraft does not rely upon its presence, that is the controller’s job.

The STCA system became operational for part of UK airspace in 1988 [6] and versions capable of coping with complex terminal control airspaces have been in operation since 1994. There are currently four en-route STCA systems in service, one covering each control centre’s jurisdiction.

STCA works by predicting the future course of aircraft and checking for conflict. Since STCA takes its input from ground radar, it has no foreknowledge of aircraft route or pilot/controller intent; it cannot predict future controller commands directing a plane to change course or level off a climb/descent. Thus, it is expected that a certain percentage of alerts will be nuisance or ‘false positive’ (FP); similarly it is possible for wanted or ‘true positive’ (TP) alerts to be missed.

STCA is a flexible system, engineered to accommodate changes in airspace usage and to

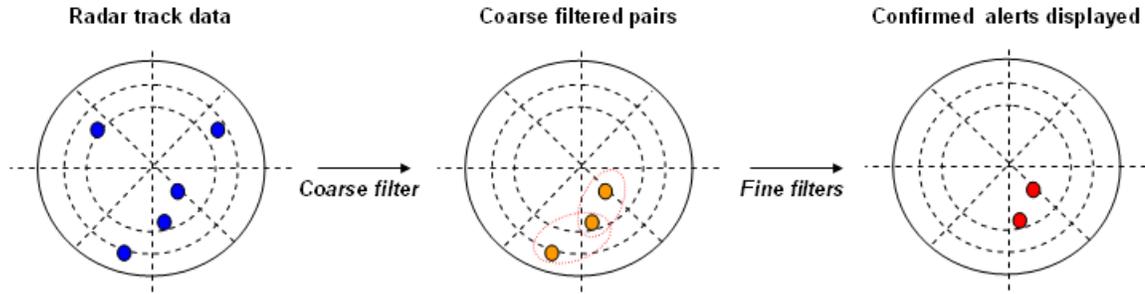


Figure 2.2: Coarse filtering of aircraft track pairs.

allow sectors to be re-defined. This flexibility results in a considerable number of control parameters that govern its behaviour and define how the software should react to aircraft in the different regions; parameters are described in more detail in section 2.3.5.

The challenge for analysts and maintenance personnel is in making informed decisions about the system’s likely performance and selecting an appropriate trade-off between ‘operationally relevant’ (TP) alerts, and those which are deemed nuisance or false (FP) [66]. Current procedure is to manually ‘tune’ the parameters in order to optimise the operating point, however this is a laborious and complex process.

2.3.1 STCA technical overview

Fundamentally, STCA deals only in *pairs* of aircraft. Every four seconds (the time taken for radar to cycle through one revolution), ground radar plots aircraft positions. The resulting tracks are passed through a *coarse* filter designed to remove any aircraft pairs that are not close enough to warrant further processing [51]. Only pairs of aircraft whose tracks are deemed sufficiently close are passed on to the ‘fine’ filters (see figure 2.2). For example, referring to figure 2.2, a series of fictional aircraft positions have been marked as blue circles. Passing these to the coarse filter results in two aircraft pairs being determined as close enough to warrant further processing by the fine filters (aircraft positions are marked in orange, pairs are circled in red). Following application of the fine filters, only one pair is determined to be in conflict (marked as red circles).

The current implementation of STCA possesses three fine filter stages, each of which can flag a potential conflict arising between a given aircraft pair (figure 2.3) [51]. The *current proximity* filter checks whether a loss of separation has already occurred. The *linear prediction* filter makes the assumption that both aircraft will continue at their current speeds and headings and calculates whether a loss of vertical or horizontal separation could arise, and the *manoeuvre hazard* filter calculates aircraft turning circles and determines separation based on the expected future position(s).

A *sliding-window* mechanism is implemented which keeps track of the number of fine filter *hits*, for each pair, that have occurred within the last few (typically 3 or 4) radar cycles [51]. Typically, three or more hits will ‘confirm’ an alert and cause it to be raised on the controller’s screen. This delay mechanism is employed to reduce spurious nuisance alerts triggered by bad

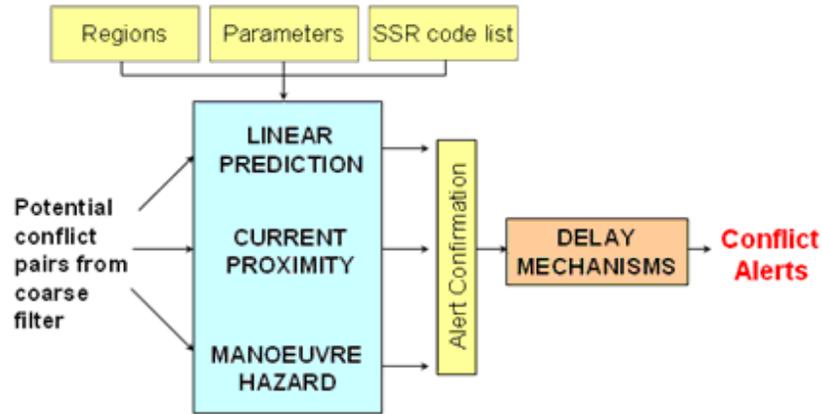


Figure 2.3: STCA fine filters. Source: NATS, 2008.

radar returns or expected aircraft manoeuvres. The current proximity filter is given higher precedence than the other filters and requires fewer confirmation hits. The size of window and number of filter hits required to trigger an alert are defined in the STCA parameters file. Each filter is controlled by a number of parameters.

In order to further reduced unwanted alerts an SSR (Secondary Surveillance Radar) code list is passed to STCA [51]. This defines aircraft call sign ranges that should be ignored by STCA; for instance, it may be desired that military aircraft be excluded as the types of combat training and flight formation manoeuvres they practise would lead to numerous STCA alerts!

2.3.2 Sectorisation

As airspace sectorisation can be complex the STCA system is set up to allow for 32 different *regions* [51] (e.g. en-route, approach, departure, stack¹ etc) in each of which the filters' operation may be tailored to suit the behaviours of aircraft in that region. Each region is specified by a given 'shape' described in latitude, longitude and vertical co-ordinates (see figure 2.4). Exclusion zones may also be defined to cover areas outside a given control centre's authority or to mark uncontrolled airspace. This is done to reduce nuisance alerts, as aircraft within these regions will be 'ignored' by the STCA system. Exclusion zones are designated 'region 0'.

It should be noted that STCA regions may not directly map to airspace sectorisations but that the regions must, as a minimum, cover all controlled airspace. Parameters are set up to control how the filters should react for each defined STCA region: There are multiple parameters defined for each of the fine filters and each parameter's value is specific to the region of airspace an aircraft pair is determined to be in.

A *combined region type* is assigned to each aircraft pair based on which regions the constituent aircraft are in [51]. This is done using a matrix: as illustrated in figure 2.5,

¹Regions situated outside of airports where aircraft circle until permission is given to land. Several aircraft may be in a stack at once flying in circular tracks separated by 1000ft vertically. Aircraft usually enter a stack at the top and leave for landing approach at the bottom. As an aircraft leaves the stack, those remaining may shift down to the track below their current position.



Figure 2.4: LTCC STCA regions. Here we see the horizontal shape co-ordinates projected down on to the ground. It should be noted that the vertical extents cannot be seen. Source: NATS, 2008.

if aircraft A was in region 2 and aircraft B in region 3, the combined region type would be 3 and thus the parameters affecting the fine filters for this aircraft pair would be those pertaining to region 3. It is the large number of regions and resulting multiplicity that leads to the vast number of STCA parameters. For example, there are ≈ 90 ‘regional’ parameters in the current STCA implementation and 17 airspace regions defined for LTCC, resulting in ≈ 1530 parameters overall.

Region	1	2	3	4	..
1	1	2	1	4	
2	2	2	3	4	
3	1	3	3	5	
4	4	4	5	4	
:					

Figure 2.5: Example STCA regions matrix.

2.3.3 Manual optimisation of STCA

NATS maintain an offline version of the STCA model that can be run on historical radar data for the purposes of analysing and monitoring system performance. Changes in airspace usage, traffic volume and operation procedures mean that STCA parameter configurations must be routinely reviewed to prevent the system from becoming out of date.

Optimisation is currently carried out on recent ‘general’ traffic comprised of $\approx 140,000$ aircraft pairs, which is about half a month’s data for LTCC and one month’s data for less busy centres such as MACC. A dataset of historical ‘Serious Encounters’ may also be processed, which contains past aircraft pair examples which led to a serious loss of separation [66]. Datasets are semi-manually categorised into ‘wanted’ and ‘unwanted’ alerts allowing an assessment to be made as to the current True Positive (TP - wanted), and False Positive (FP - unwanted) rates of the system, along with the warning time given to wanted alerts. Each aircraft pair is assigned one of four categories:

CAT1 - Risk of collision

STCA alert required (wanted). Serious loss of separation occurred or was likely to occur in the next 30 seconds (7-8 radar cycles) but was averted by a late manoeuvre.

CAT2 - Significant loss of separation

STCA alert desirable (wanted). Significant loss of separation occurred or was likely to occur in next 60-80 seconds (15-20 radar cycles) but was averted by a manoeuvre other than a complete level off, and there was risk of collision. A ‘level off’ is where an aircraft that was previously climbing or descending levels out and maintains a given altitude. Level offs occur frequently near airports as aircraft queuing to land descend the stacks in a series of levels, usually separated by 1000ft vertically.

CAT3 - Level off with risk

STCA alerts understandable (but unwanted). A level off manoeuvre (at least 700ft vertical separation) averts a significant loss of separation. That is, the aircraft probably always was going to level off; checking the aircraft’s selected flight level (if available) can confirm this. As STCA has no knowledge of pilot intention it cannot accurately predict whether an aircraft will level off or ‘bust’ through leading to a potential conflict. Errors in predicting level off lead to nuisance alerts, so CAT3s are marked as unwanted.

CAT4 - Nuisance

STCA alerts undesirable (unwanted). Significant loss of separation is never likely to occur, there is no risk of collision.

CAT5 - Bad data

Invalid for optimisation. A ‘CAT5’ can be assigned to aircraft pairs. This is used to indicate incomplete track data or erroneous data points making aircraft appear to be in conflict when they are not. Such pairs are ignored during optimisation, as wanted/unwanted status cannot be assigned.

A categorisation tool (Category_V3²) is used to aid the process by assigning CAT4s and CAT2s automatically, however manual review of its output is required in order to assess which pairs should be re-assigned to CAT1 and CAT3 status.

Categorised datasets are passed through an off-line model of STCA called Conflict Alert Management Performance Analysis Package (CAMPAP) [7, 1]. The performance of STCA in correctly identifying each category in *each airspace region* is output. Over the last 12 years, skilled staff at NATS have been incrementally modifying the parameters that control STCA and assessing the changes in system performance by repeated evaluation using CAMPAP. However, with modifications to the STCA technology and increasingly complex airspaces this task has become ever more demanding.

2.3.4 Enhanced Short Term Conflict Alert (ESTCA)

ESTCA is NATS’ next generation of STCA system [66]. It improves on the original (Issue 4) specification [51] by adding an additional fine filter designed to predict the lateral path of aircraft holding in stacks [52]. Additionally, it allows downlinked Selected Flight Level (SFL - Mode-S data) information to be used in reducing CAT3-type nuisance alerts and predicting level busts. There are also a series of minor improvements to existing logic. ESTCA is currently undergoing testing and implementation. Of course, the consequence of further system improvements is that there are more parameters (see figure 2.1) that require tuning.

However, as far as any automated optimisation process is concerned both STCA and ESTCA can be treated as a ‘black-box’ that classifies aircraft pairs into alerts or non-alerts, the only difference being that ESTCA introduces a new set of parameters to be adjusted.

2.3.5 STCA parameters

Since aircraft in different region types tend to have different types of flight behaviour, separate parameter sets are used for each one of the region types. The particular parameter set used for classifying a track pair therefore depends upon the STCA region types of the two aircraft. As previously stated in section 2.3.2, it is the large number of regions and resulting multiplicity that leads to the vast number of STCA parameters.

As an example, the current number of parameters for the MACC STCA system is as follows:

Number of named parameters:	126
Number of optimisable named parameters:	78
Number of optimisable regions:	13
Total number of optimisable parameters:	942

These values are based on coarse filter, standard manoeuvre time, alert severity code criteria and optional ‘off-line’/non-operational parameters *not* being eligible for optimisation. (See appendix A for an example of STCA parameter file layout.)

²Software designed to perform preliminary filtering of aircraft. Category_V3 coarsely divides aircraft into likely conflict pairs and benign encounters.

(General parameters)							
GROUNDELEVATION	500	(ft)					
HIGHLEVEL	29201	(ft)					
CYCLE	4	(cycles)					
FLZTOL	200	(ft)					
...							
(Fine filter region dependant values)							
FFDIVFAST	120	120	120	120	120	...	(knots)
FFDIVSEPL	1.0	1.0	1.0	1.0	1.0	...	(NM)
...							
(Linear prediction filter)							
LPLDA	0.75	0.75	0.5	0.75	0.5	...	
LPVDA	0.75	0.75	0.5	0.75	0.75	...	
...							

Figure 2.6: Example of regional/non-regional parameter values. Multi-column entries are specified for regional parameters (one for each region).

MACC is divided into 13 of these different region types. There are 126 ‘named’ parameters pertaining to the three fine filters, the coarse filter and other control logic; of these ≈ 90 are multi-regional, that is to say the value assigned is dependant on the combined region type of aircraft. In total then there can be up to $(13 \times 90) + 36 = 1206$ tunable parameters for MACC. However, in reality only 78 of the ‘named’ parameters are routinely adjusted, which, taking into account the regional multiplicities, leads to a total of 942 tunable values. Non-optimisable parameters are often those that are dictated by regulation, for instance the minimum aircraft separation required in UK airspace is 5NM horizontal, 1000ft vertical.

To clarify, figure 2.6 shows examples from a parameter file. The named parameters GROUNDELEVATION (geographic elevation of ground above sea level), HIGHLEVEL (lower boundary of high level airspace), CYCLE (fine filter cycle period) and FLZTOL (tolerance margin for conformance to flight level) are all non-regional, that is they are not dependant on aircraft region. The named parameters FFDIVFAST (lateral speed threshold for fast diverging aircraft), FFDIVSEPL (minimum lateral separation for fast diverging), LPLDA (lateral divergence allowance) and LPVDA (vertical divergence allowance) are region-dependant and thus multiple values must be specified, one for each possible combined region type [51] (here only 5 regions have been shown for brevity). It should be noted that in a large number of cases identical values are assigned for each regional parameter.

It might be expected that regional parameters would be independent of one another. However, as will be discussed in chapter 4 section 4.3, the cross regional interactions between the combined region type matrix and the alert confirmation sliding window mean that there is some interference between them. This means that regional parameters cannot truly be adjusted independently.

	STCA	ESTCA
Number of named parameters:	126	257
Number of optimisable named parameters:	89	224
Number of optimisable regions:	17	17
Total number of optimisable parameters:	1417	3863

Table 2.1: Optimisable LTCC STCA and ESTCA parameters.

Parameter values include both floating point and integer values over many varying ranges and measured in a variety of units, examples of which are ‘vertical closing rate threshold’ (1600 to 3000 ft/min), ‘lateral miss distance criterion’ (1.7NM to 2.5NM) and ‘total reaction time before lateral manoeuvre’ (25 to 55 seconds). In addition, discrete valued or ‘categorised’ parameters may be present e.g. ‘level test’ may only be set to 1 or 0 (True or False). Discrete valued parameters may also have multiple regional values defined.

Table 2.1 compares how the number of parameters for LTCC will increase with the introduction of ESTCA. We can see that the number of optimisable parameters will more than double. This is largely due to previously non-regional parameters having been split into multiple values for each region. This was done to increase the flexibility of the system.

2.3.6 The offline STCA environment

The offline STCA model (‘Conflict Alert Management Performance Analysis Package’ - CAMPAP) [7, 1] currently resides on a DEC Alpha AXP system equipped with three CPUs. Although originally established for the sole purpose of maintaining STCA, the system is under increasing demand for data processing from other users. The software is written in an OpenVMS OS-specific version of PASCAL, thus porting the code to other platforms for the purposes of our work would not be a viable option. Irrespective of whether it were possible, making a duplicate is inadvisable, as in doing so we would risk introducing changes that make the offline model different from the operational version.

Figure 2.7 shows the linear relationship between the number of aircraft pairs in an STCA dataset and the CPU time taken to process it using CAMPAP. Given a halved dataset of $\approx 70,000$ pairs we can see that it takes in the region of 3 mins 30 secs to evaluate one STCA parametrisation (even longer on the new ESTCA system). Thus, any iterative automated optimisation process could take literally weeks to attain usable results. While this is CPU time and not person-hours, anything that can be done to reduce optimisation time will clearly improve productivity and reduce the impact on system resources.

We emphasise the expensive nature of this optimisation problem and highlight that, whilst we endeavour to conduct a comprehensive investigation, time and computational resource is a limiting factor.

See appendix C for further details of the offline STCA architecture.

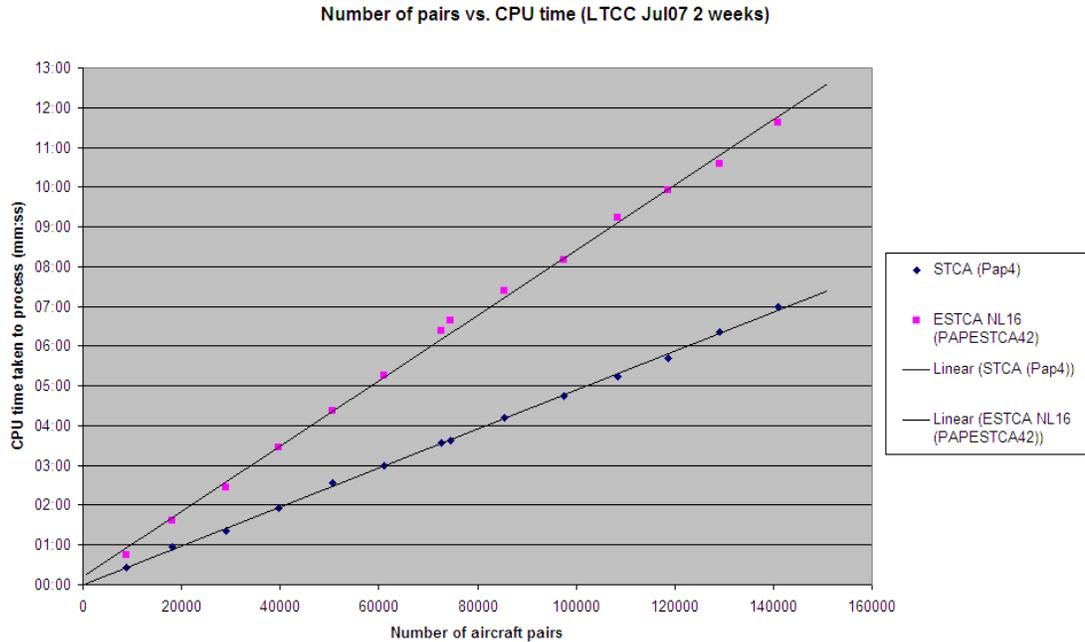


Figure 2.7: STCA & ESTCA, Number of aircraft pairs vs. CPU time.

2.4 Automated optimisation of multiple objectives

In ‘real world’ problems, it is often the case that optimisation of multiple objectives will not result in a single ‘optimum’ solution that is wholly better than all others. That is, where objectives are conflicting, and frequently they are, a trade-off curve or surface may be observed; an improvement in one objective will result in degradation of another, as is the case with STCA. Of course, if objectives are not in conflict then it is expected that convergence to a single optimal or near optimal solution should be possible, however, non-competitive objectives are comparatively rare in industrial problems (see [17] for examples).

In section 2.3.3 optimisation of STCA was presented as a laborious manual process. The following sections will introduce evolutionary algorithms for automatic optimisation. We will cast optimisation of STCA as a multi-objective problem suitable for automated exploration and describe how the Receiver Operating Characteristic (ROC) of an STCA system might be generated.

		Expected value (true class)		Total:
		p	n	
Classifier assignment	p'	True Positive	False Positive	P'
	n'	False Negative	True Negative	N'
Total:		P	N	

Figure 2.8: Classifier confusion matrix.

2.4.1 ROC analysis

Traditionally ROC analysis has its roots in signal processing [27]. However, in recent years its relevance to machine learning, especially amongst tasks with class skew and unequal classification costs, has been increasingly appreciated [36]. An ROC is essentially a trade-off curve, plotting True Positive Rate (TPR) against False Positive Rate (FPR) for a given classifier.

True positive rate – the rate at which members of the positive class, P , are correctly identified by a classifier. In some domains true positive rate may also be called the ‘recall’ or ‘sensitivity’ of a classifier. In information retrieval for instance, ‘recall’ refers to the number of relevant documents that were retrieved by a search but does not account for how many irrelevant documents were also included.

$$\text{TP rate} = \frac{\text{Positives correctly classified}}{\text{Total positives}} = \frac{TP}{P} \tag{2.1}$$

$$\text{recall} = \text{sensitivity} = \text{TP rate} \tag{2.2}$$

False positive rate – the rate at which members of the negative class, N , are incorrectly identified by a classifier as belonging to the positive class, P .

$$\text{FP rate} = \frac{\text{Negatives correctly classified}}{\text{Total negatives}} = \frac{FP}{N} \tag{2.3}$$

For any given classifier a two by two ‘confusion’ matrix may be constructed to illustrate the relationship between TPR, FPR and other common metrics (see figure 2.8). The confusion matrix summarises the performance of a classifier.

Referring to figure 2.8, we adopt the notation P and N to represent the total number of instances in the positive and negative classes respectively; it is worth noting that what constitutes as the ‘positive class’ is defined by a decision maker (i.e. ourselves). p' and n' are used to represent the positive and negative class assignments as determined by a given classifier. Thus, the major diagonal (top-left to bottom-right) of the confusion matrix lists the number of instances correctly identified by the classifier. The off-diagonal (top-right to bottom-left) represents the number of incorrectly identified instances. Note that if the TPR

and FPR are known, then the other quantities in the confusion matrix are easily calculated. Common metrics of classifier performance are [36]:

Precision – used in information retrieval to measure relevancy, i.e. how many of the results returned from a search were relevant. However, this measure does not account for documents that should have been included but were not.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.4)$$

Accuracy – a measure of a classifiers ability to correctly separate classes.

$$\text{Accuracy} = \frac{TP + TN}{P + N} \quad (2.5)$$

Specificity – a measure of the true negative rate, i.e. the rate at which members of the negative class, N , are correctly identified by a classifier.

$$\text{Specificity} = \frac{TN}{FP + TN} = 1 - \text{FP rate} \quad (2.6)$$

F-measure – the f-measure or f-score is the weighted average of the precision and recall (results are in the 0–1 range). In information retrieval, the f-score measures the effectiveness of a search.

$$\text{F-measure} = \frac{2}{1/\text{Precision} + 1/\text{Recall}} = \frac{2TP}{P + P'} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.7)$$

AUC – the area under curve can be used as a scalar measure of classifier performance in respect to a given ROC curve. The area under an ROC curve is equal to the probability that a classifier will rank a randomly chosen positive instance, p , higher than a randomly chosen negative, n . The AUC may be estimated by rectangular or trapezoidal approximations (see figure 2.9) or via the Mann-Whitney-Wilcoxon test – see [50]. It is essentially the integral of an ROC function. Throughout our trials we use the rectangular method of calculation, which is a conservative estimate of the AUC, corresponding to the area dominated by the attainment surface.

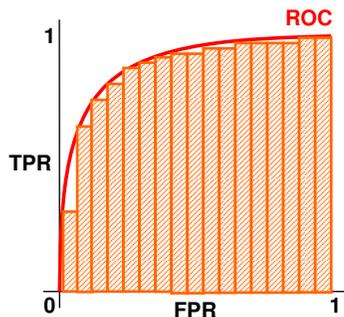


Figure 2.9: AUC estimation. Note how at steeper gradients the orange rectangles fail to adequately approximate the ROC (red line), this can be resolved by increasing the resolution at which rectangles are generated.

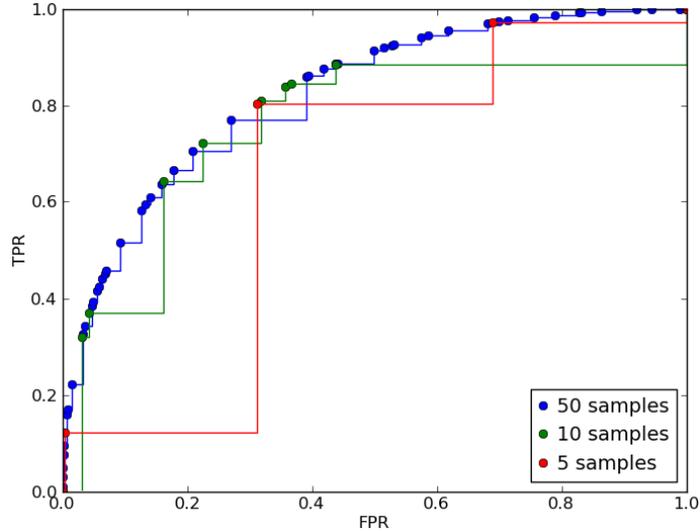


Figure 2.10: Fictional ROC for a hard classifier sampled at various resolutions.

There exist two main types of classifier; discrete (or hard) classifiers and thresholded/probabilistic (or soft) classifiers. The former output only a class label (as is the case with STCA), whilst the latter output a continuous value that may be thresholded, such as those based on a neural net or Bayesian framework that output a probability of class assignment. The two types require slightly different approaches to ROC construction.

In the case of soft classifiers a threshold value may be used as a meta-parameter, sweeping out the arc of the ROC by varying the threshold and thus tuning the classifier’s segregation of one class from another. The ROC curve shows the performance as the misclassification cost rate varies, or rather as the bias towards misclassifying one class over another is varied. Clearly in some problem domains this is beneficial as the cost of misclassifying one class (e.g. ‘cancerous’) is higher than misclassifying the other (e.g. ‘benign’).

An alternative methodology is required for hard classifiers as each discrete output can only generate one single confusion matrix and thus a single TP/FP point in ROC space. Essentially we have to sample several different classifier configurations, plotting the output in TP/FP space in an attempt to build up an approximate ‘description’ of the true ROC. The more ‘optimal’ solutions we sample the greater the resolution or accuracy of the resulting curve (see figure 2.10). However, regardless of whether a classifier is hard or soft, as Fawcett remarks; “any ROC curve generated from a finite set of instances is actually a step function, which approaches a true curve as the number of instances approaches infinity” [36], that is, the number of class instances (in the case of STCA, aircraft pairs) evaluated will have a direct impact on the refinement of the resulting ROC.

2.4.2 Automated optimisation of STCA

When optimising STCA we would like to maximise the number of wanted alerts and minimise the number of nuisance alerts, however, we cannot improve on one with out adversely affecting the other. Therefore, we must construct a trade-off curve. In ‘tuning’ parameters, it is possible

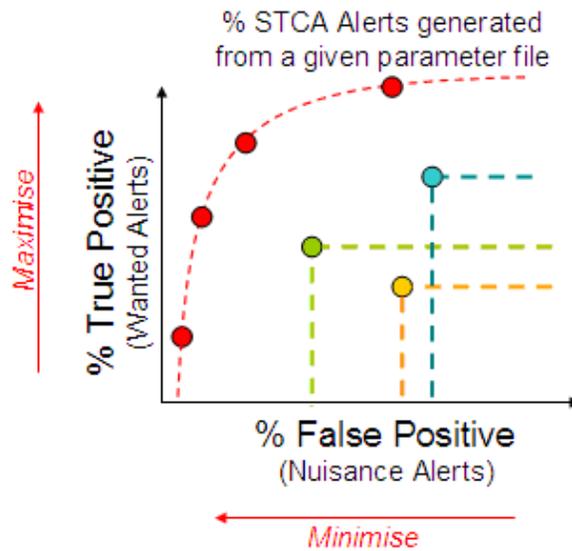


Figure 2.11: Dominance example.

to plot the corresponding wanted/nuisance alerts generated as a true positive rate against the false positive rate. Each separate set of parameter values thus generates a single point in the (FP, TP) plane. This allows direct comparison of parametrisations in terms of their dominance (which is ‘better’ in terms of both the TP and FP rate) over one another; i.e., a parametrisation that has a lower TP rate and higher FP rate than another is said to be dominated and should be rejected. A trade-off curve can be created by plotting mutually non-dominating parametrisations against one another. Through a process of repeated, ‘intelligent’ perturbation of parameter values, the optimal set of mutually non-dominating points can be constructed. This is called the *Pareto Front*.

For instance, the green point in figure 2.11 dominates the yellow. That is to say it is ‘better’ in respect of both the TP and the FP rates. The green and blue points are said to be mutually non-dominating, as green generates fewer false positives but also fewer true positives, however the blue produces more wanted alerts but also more false positives. The red points in figure 2.11 are the Pareto front. This is the optimal set of mutually non-dominating points, which are not dominated by any other conceivable parametrisation.

The Pareto front is essentially a cost/benefit representation of the effect of system tuning; a plot of the trade-off associated to increasing true positive or decreasing false positive classifications. Inherently, on the Pareto front an increase in TP will always correspond to an increase in the resulting FP rate and similarly, attempting to decrease FP will reduce TP.

The automated optimisation process is fundamentally a directed form of ‘trial and error’. By creating software to make small incremental changes to those STCA parametrisations found to be non-dominated by any other parametrisations found so far and evaluating the resulting performance, over many generations a superior set emerges, pushing the estimate of the Pareto front forwards. In addition, the concept of ‘dominance’ can easily be extended to multiple dimensions (as opposed to just the TP/FP plane) allowing optimisation of, for

instance, the alert time given to controllers to correct a potential aircraft conflict [35].

In the following section the concept of dominance and Pareto optimality will be presented more formally.

2.4.3 Application of ROC analysis and Pareto Optimality

The notion of ‘Pareto Optimality’ [14, 78, 84, 15] is often used to define what ‘optimal’ means in the context of multiple objectives. As we have discussed, in the presence of conflicting objectives there is often no one overall optimal solution. Rather, there exists a trade-off surface whereby no one solution entirely better (dominates) another in all objectives; an increase in the fitness of one objective cannot be made without decreasing the fitness in at least one other. Thus, a set of globally *non-dominating* solutions may be derived, designated the *Pareto Front* after Vilfredo Pareto an Italian economist who first formalised the concept. A set of solutions is said to be Pareto optimal if no further improvement can be made in one or more objectives without degrading the solutions in another.

In the context of STCA, we require a method of automatically locating the set of optimal solutions which lie along the ROC curve; the Pareto Front. As noted above, we may regard the STCA system as a classifier, $g(\mathbf{x}; \boldsymbol{\theta})$, which gives an estimate of the probability that a feature vector \mathbf{x} (for STCA the radar tracks of a pair of aircraft) belongs to one of two classes. We assume that the classifier depends upon a vector of adjustable parameters $\boldsymbol{\theta}$, and we denote by $TP(\boldsymbol{\theta})$ the classifier’s true positive classification rate, while the false positive rate is denoted by $FP(\boldsymbol{\theta})$.

If the costs of an incorrect classification were known it would be straightforward to calculate the expected cost for any particular parameter and data set [24]. It would then be possible to adjust the parameters to minimise the expected cost. However, this procedure requires accurate specification of the misclassification costs which are seldom known accurately. The Receiver Operating Characteristic (ROC) curve displays the trade-off between true and false positive rates as the ratio of misclassification costs is varied for a particular fixed set of parameters. As the misclassification cost ratio is varied a non-decreasing ROC curve in the (FP, TP) plane is obtained for any particular fixed set of parameters, and different, possibly intersecting, ROC curves are obtained for different parameters. With the ROC curves on hand the user can select the operating point with a full knowledge of the possible trade-offs involved.

Of course, all these measures based upon the ROC curve require knowledge of the ROC curve, which hitherto has been unavailable for the STCA system. In this thesis we show how multi-objective evolutionary algorithms may be used to derive the ROC curve for the STCA system optimised over all possible parameter values. That is, we seek to discover the set of parameters that simultaneously minimise $FP(\boldsymbol{\theta})$ and maximise $TP(\boldsymbol{\theta})$.

A general multi-objective optimisation problem seeks to simultaneously extremise D objectives:

$$y_i = f_i(\boldsymbol{\theta}), \quad i = 1, \dots, D \quad (2.8)$$

where each objective depends upon a vector $\boldsymbol{\theta}$ of P parameters. For the purposes of notation, it is convenient to assume that all the objectives are to be minimised, so for the STCA system we might minimise the pair of objectives $(-TP(\boldsymbol{\theta}), FP(\boldsymbol{\theta}))$. The multi-objective optimisation problem may be conveniently expressed as:

$$\text{minimise } \mathbf{y} = \mathbf{f}(\boldsymbol{\theta}) = (f_1(\boldsymbol{\theta}), \dots, f_D(\boldsymbol{\theta})) \quad (2.9)$$

where $\boldsymbol{\theta} = (\theta_1, \dots, \theta_P)$ and $\mathbf{y} = (y_1, \dots, y_D)$. Note that the parameters may also be subject to some constraints, such as being within a safe operating range.

When faced with only a single objective an optimal parametrisation is one which minimises the objective given the constraints. However, when there is more than one objective to be minimised parametrisations may exist for which performance on one objective cannot be improved without sacrificing performance on at least one other. Such parametrisations are said to be *Pareto optimal* [14, 78, 76, 42] and the set of all Pareto optimal parametrisations is said to form the Pareto set.

The notion of *dominance* may be used to make Pareto optimality clearer. A parameter vector $\boldsymbol{\theta}$ is said to *strictly dominate* another $\boldsymbol{\phi}$ (denoted $\boldsymbol{\theta} \prec \boldsymbol{\phi}$) if and only if all the objectives corresponding to the parameter $\boldsymbol{\theta}$ are no worse than those obtained with $\boldsymbol{\phi}$ and at least one objective is better, that is:

$$f_i(\boldsymbol{\theta}) \leq f_i(\boldsymbol{\phi}) \quad \forall i = 1, \dots, D \quad \text{and} \quad f_i(\boldsymbol{\theta}) < f_i(\boldsymbol{\phi}) \quad \text{for some } i. \quad (2.10)$$

Less stringently, $\boldsymbol{\theta}$ weakly dominates $\boldsymbol{\phi}$ if the objectives corresponding to $\boldsymbol{\theta}$ are all at least as good as those corresponding to $\boldsymbol{\phi}$:

$$f_i(\boldsymbol{\theta}) \leq f_i(\boldsymbol{\phi}) \quad \forall i = 1, \dots, D. \quad (2.11)$$

A set A of decision vectors is said to be a *non-dominated set* if no member of the set is dominated by any other member:

$$\boldsymbol{\theta} \not\prec \boldsymbol{\phi} \quad \text{for all } \boldsymbol{\theta}, \boldsymbol{\phi} \in A. \quad (2.12)$$

A solution to the minimisation problem (2.9) is thus *Pareto optimal* if it is not dominated by any other feasible parametrisation, and the non-dominated set of all Pareto optimal parametrisations is the Pareto front. Recent years have seen the development of a number of evolutionary techniques based on dominance measures for locating the Pareto front; see section 2.4.5 and [22, 84, 14, 76, 42] for recent reviews. We will implement a similar iterative approach whereby repeated perturbation and evaluation for dominance leads to a population of increasingly good parametrisations and ultimately the Pareto-optimal front.

The iterative nature of metaheuristic algorithms permits rapid convergence on such optimal or near optimal solutions. In the next sections we will discuss how metaheuristic algorithms can be applied to solving combinatorial optimisation problems.

2.4.4 Tabu search

A tabu search is a metaheuristic approach to optimisation in which varying configurations (θ) are generated via incremental or stepped changes to parameter values from a given starting composition (see [45] for further details). It is a *local* search method, generating a ‘neighbourhood’ of other possible solutions from a given starting parametrisation. At each iteration the algorithm selects the ‘best’ member of the current neighbourhood and takes it as the starting solution for generation of a new neighbourhood. To prevent backtracking over previously visited solutions the algorithm maintains a ‘tabu’ list. This keeps track of previous ‘moves’, preventing cycling and thus helping to avoid getting stuck in local optima. So that the length of the list does not grow inhibitorily large, entries are removed after a given number of generations or when some other predefined criteria is met, allowing blacklisted solutions to be reconsidered.

A method of optimising STCA via tabu search was suggested by Beasley *et al.* [6]. Their emphasis was on evaluating new logic/functions potentially to be added into the STCA base code, in order to do this parameter optimisation was required. A composite objective function was used to generate a single fitness score for each new STCA parametrisation generated. This was based on weighted summation of ‘alerts gained’, ‘alerts lost’, and warning time (cycles) lost or gained, meaning this method implicitly requires comparison to an accepted ‘base case’; the NATS manually tuned operating point.

Limitations were found to exist with this approach. It was discovered that under certain circumstances the introduction or activation of new logic would initially lead to a degradation in performance (an inferior fitness score compared to the base case), implying that the tabu search had become stuck in local optima. The authors suggest that with manual analysis an appropriate starting point may be constructed to avoid such a predicament, however, the requirement of manual intervention is less than ideal for production systems. Indeed, the necessity to always initialise from the NATS current optimum, manually corrected or otherwise, has another drawback; using this method to tune an STCA system on a newly reconfigured airspace or major overhaul of logic, for which no base case exists, becomes problematic. There is no mention of how the algorithm might perform when starting from a totally randomised parametrisation.

In addition, in its most basic form a tabu search seeks only a single optimum solution. That is, the composite fitness function requires prior knowledge of appropriate costs (weights) associated with each objective. Instead, we would ideally like to present personnel with an ROC curve and let them decide the appropriate system trade-off from this; not require the decision to be made *a priori*. Overfitting, whereby the ‘optimum’ parametrisation becomes too adapted to the training data and does not generalise well to ‘unseen’ instances, was also cited as a concern. This, however, is a potential problem for all machine learning techniques.

Due to these limitations we do not examine this methodology any further. In [37, 35] Everson and Fieldsend propose an alternative metaheuristic approach to automating optimisation of STCA; evolutionary algorithms.

2.4.5 Multi-objective evolutionary algorithms

The term ‘Evolutionary Algorithm’ (EA) is used to refer to a subset of heuristic algorithms inspired by processes observed in the natural world. In nature, the mutation and crossover of parent genotypes leads to new phenotype variants in children. Those children possessing a resultant natural advantage, have a higher chance of surviving to the next generation and passing on their genes. EAs attempt to model this behaviour by iteratively improving candidate parametrisations that are to be optimised over successive generations. However, they are possibly more akin to selective breeding than natural evolution because we essentially apply a selective pressure such that traits we find desirable will be promoted.

The use of EAs as an optimisation technique has grown in popularity, part in thanks to their greater resilience to being trapped in local optima and natural extension into multiple, potentially conflicting, objectives. Modern Multi-Objective Evolutionary Algorithms (MOEAs) integrate techniques for archiving a set of optimal or near optimal solutions, allowing approximation to an ROC curve in a single run. Over the last ≈ 20 years many variations of MOEA have been proposed, and extensive reviews of such algorithms can be found in [42, 14, 78, 76, 84, 15]. Initial experiments by Everson and Fieldsend [35] suggest this approach to optimisation of STCA to be productive. In the following pages a brief overview of perhaps the most significant variants are presented, outlining the advancement from early concept to the present ‘standard’. We will introduce the basic concepts behind MOEAs, before illustrating how such an algorithm might be applied to STCA in section 2.4.2.

Basic EA outline

In its simplest form an Evolutionary Algorithm consists of four main steps; selection, perturbation, evaluation and update, of a population of individuals (see figure 2.12). A population of candidate solutions is maintained from which a member is selected (copied), modified (has ‘genetic operators’ applied) and evaluated (has a fitness function applied). Based on the assigned fitness, a decision is then made as to whether the new child solution should enter the population and if so, whether any current members of the population should be discarded. From this simple iterative procedure a surprising number of algorithmic variations may be devised.

The fitness function determines population ranking, it directs the search towards optimal solutions. A fitness function should account for each objective to be optimised, ideally without requiring *a priori* knowledge of associated costs (as these may be unknown). Good selection/update functions should promote fit individuals whilst maintaining chromosome (parametrisation) diversity in a population. If a population is not diverse, the search may become stuck in local optima. Conversely, if a population is too diverse it may dilute selective

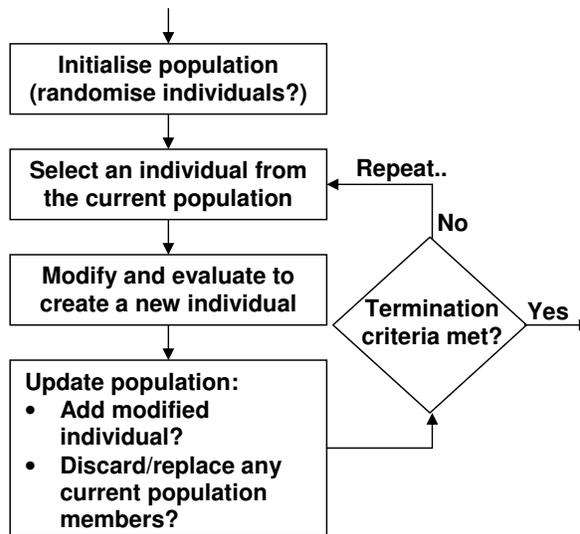


Figure 2.12: EA outline.

pressure leading to unnecessarily slow convergence to a global optimum.

Perturbation refers to the process by which new child solutions are derived from their respective parents. A selected parent is modified by applying genetic operators from two different categories; mutation and cross-over. Mutation operators modify individual parameter values. That is, they increment or decrement a value by addition or subtraction, either uniformly or based on some other distribution. Cross-over operators ‘splice’ two or more chromosomes together, taking a subset of parameters from one parent and replacing the corresponding subset in another.

It should be noted that the term ‘Genetic Algorithm’ (GA) is often used to describe a subset of Evolutionary Algorithms. GAs are conceptually similar to EAs however, they are usually characterised by their method of encoding solutions as binary strings rather than real-valued parametrisations and use of cross-over.

The rest of this discussion concerns how fitness may be derived from multiple objectives, assignment schemes may be divided into two distinct subsets; Pareto and non-Pareto based techniques.

Non-Pareto based methods

Many early examples of EA implemented an aggregation or composite method of determining and comparing solution fitness [42, 74]. That is, objectives are added together to give a single scalar-value ‘score’. Weightings can be used to adjust the perceived importance of one objective relative to another (weighted-sum) or, if a specifically desired measure or cost is known, the distance of each objective to its given ‘goal’ may be used to calculate fitness (goal attainment). These two methods are conceptually similar, however, weighted-sum requires that only the *relative* importance of objectives need be known [74]. The simplicity of aggregation approaches allows easy construction of an overall ‘goal’ function, used to direct an EA’s search. Here we denote the aggregate goal function $f(\theta)$. Approaches to aggregation

include weighted-sum, Min-Max/weighted Min-Max, the Global Criterion Method, L_p -norm and ϵ -constraint. These are defined as follows:

Weighted-sum [43] – in the weighted-sum method objectives f_1, \dots, f_D , weighted by w_1, \dots, w_D are added together to give a scalar fitness value;

$$f(\boldsymbol{\theta}) = \sum_{i=1}^D w_i f_i(\boldsymbol{\theta}). \quad (2.13)$$

Some authors have used a weighted-product approach based on logarithms [75], however this is seen to be equivalent to a weighted-sum approach.

Min-Max [43] – the Min-Max method minimises the maximum difference between D objectives, f_1, \dots, f_D , and the user specified goals g_1, \dots, g_D ;

$$f(\boldsymbol{\theta}) = \max_{i=1, \dots, D} f_i(\boldsymbol{\theta}) - g_i. \quad (2.14)$$

In practice a weighting of the difference between objectives and goals may be applied [43] (weighted Min-Max);

$$f(\boldsymbol{\theta}) = \max_{i=1, \dots, D} w_i (f_i(\boldsymbol{\theta}) - g_i). \quad (2.15)$$

Global Criterion Method – the Global Criterion Method, also known as the classical Min-Max method [16], minimises the *relative* distance of objectives to a target vector (the ‘global criterion’);

$$f(\boldsymbol{\theta}) = \sum_{i=1}^D \left(\frac{g_i - f_i(\boldsymbol{\theta})}{g_i} \right)^p \quad (2.16)$$

where p controls the bias towards minimising the objective, f_i , with the highest value. It is usually set as 1 or 2 [16].

L_p -norm [16] – L_p -norm is calculated as the sum of the differences of objectives to goals. Again, p is used to control the bias towards minimising the objective with the highest value;

$$f(\boldsymbol{\theta}) = \left(\sum_{i=1}^D |g_i - f_i(\boldsymbol{\theta})|^p \right)^{1/p}. \quad (2.17)$$

ϵ -constraint [28] – in the ϵ -constraint method a single objective is minimised, constrained by the remaining objectives;

$$f(\boldsymbol{\theta}) = \min f_i(\boldsymbol{\theta}) \quad (2.18)$$

subject to $f_j(\boldsymbol{\theta}) \leq \epsilon_j \quad j = 1, \dots, D \quad j \neq i$, where $\epsilon \in \mathbb{R}^D$.

However, as with tabu search, there are disadvantages inherent in such schemes; an aggregating function will yield only a single ‘optimal’ solution, multiple-runs using varied weights may be required to generate a Pareto Front and correct calibration of the weights requires a prior in depth knowledge of costs associated to each objective.

Lexicographical ordering [9] was proposed as an alternative to aggregation functions. Here, objectives are optimised one at a time based on order of importance. That is, the ‘optimal’ solution resulting from optimisation of the first (deemed most important) objective is taken as the starting point for optimisation of the second (next most important) objective. Any perturbations that improve the second objective must do so without degrading performance on the first (or they will not be accepted), and so on for all remaining objectives, each in turn. However, this method still requires a prior knowledge or estimation of objective significance.

Schaffer’s Vector Evaluated Genetic Algorithm (VEGA) [71] is perhaps the earliest example of a multi-objective evolutionary algorithm. VEGA’s novelty lay in its selection scheme. At each iteration VEGA generated a set of sub-populations, each representing the fittest solutions for a given objective (proportionally selected). These sub-populations were then ‘shuffled’ together prior to mutation/cross-over. That is, the same solution may be found to be most ‘optimal’ in more than one sub-population *and* represented more than once within that sub-population (due to proportional selection), thus on re-shuffling there is a greater chance that it will be selected for mutation/crossover in the next generation. However, Richardson *et al* [69] noted that the method of shuffling proportionally selected sub-populations is essentially equivalent to a linear combination of weighted objectives (dependent on the distribution of the population), and thus VEGA is subject to the same limitations as aggregation techniques.

Perhaps the most limiting factor of many early aggregation techniques was illustrated by Das and Dennis [21]. They show that, whilst it is possible to reach alternative regions of a Pareto Front by varying weights associated to a weighted-sum aggregation function, if the front is non-convex then there will be areas that cannot be accessed, even though a feasible solution may exist. In addition, an even spacing of weight intervals does not necessarily guarantee an even spacing of Pareto optimal solutions across the front. That is, there may be some sections where solutions, obtained through use of an aggregating function, are highly clustered and others in which there are none present at all.

Whilst, goal attainment or target vector approaches such as Min-Max and the Global Criterion Method do not suffer from this limitation, that is, concave portions of the Pareto front *are* attainable, they are limited by the necessity to provide goal objectives *a priori*. Additionally, acquisition of the Pareto front may require multiple runs or integration with a technique that permits its generation in a single run of the algorithm (see for instance Fonseca and Fleming’s MOGA [41], discussed in the next section or Hughes’ MSOPS [55]).

Due to such deficiencies, researchers moved away from single-objective aggregation approaches to multi-objective problems, towards a Pareto-dominance based methodology. Modern EAs allow simultaneous generation of an entire Pareto Front in a single run, more elegantly handling situations with unknown cost and efficiently generating a trade-off curve.

We note recent interest in multiple weighted aggregation for *many* objective optimisation; see MSOPS-II [56], which attempts to address the necessity for *a priori* knowledge of search direction(s) by implementing a method of automatic target vector generation. However, Pareto-dominance methods, which we discuss in the next section, have been shown to be good for optimising 2, 3 or 4 objectives, as is the case for STCA.

Pareto-dominance based methods

It was Goldberg’s criticism of the VEGA algorithm [47] and subsequent suggestion of an alternative Pareto based ranking scheme that dictated the future direction of work in the field of evolutionary algorithms. His ideas directly inspired the creation of algorithms such as MOGA [41], NSGA [74] and NPGA [53]. Goldberg proposed a method of ranking solutions based on levels of non-domination. That is, individuals found not to be dominated with respect to the rest of the population would be assigned rank 1; disregarding all members of rank 1, a second set of non-dominated solutions would be identified and assigned rank 2, and so on. To prevent convergence to a single solution a niching mechanism such as fitness sharing was suggested [15]. The idea was to maintain a population of non-dominating solutions, representing an estimated Pareto optimal front, that over many evolutionary generations would better approximate the true front.

In Fonseca and Fleming’s Multi-Objective Genetic Algorithm (MOGA) [41], solution fitness is initially assigned as 1 plus the number of members by which the solution is dominated. Thus, non-dominated solutions on the current estimated Pareto Front are assigned rank 1 (and so are sampled for selection at the same rate as one another [15]) and dominated members of the populace are assigned a fitness proportional to surrounding population density (and so are sampled based on how well ‘described’ that region of the trade-off is). This notion of fitness was then extended to allow incorporation of a goal attainment strategy; Fonseca and Fleming suggested that a ‘decision maker’³ is less interested in the full extents of the Pareto Front and more concerned with the specific portions that contain a ‘good compromise’ solution. Thus, the inclusion of goal based fitness was designed to ‘zoom in’ on only relevant areas of the Pareto set. Non-dominated solutions could be compared to predefined goal values for each objective, those non-dominated members that satisfy the goals being considered superior to those that do not. This meant that, whilst the MOGA did not suffer from areas of the Pareto Front being unreachable (as with weighted-sum aggregation), it was still disadvantaged by requiring *a priori* knowledge of objective costs.

The Non-dominated Sorting Genetic Algorithm (NSGA) proposed by Srinivas and Deb implemented “a ranking selection method . . . to emphasize good points and a niche method . . . to maintain stable sub-populations of good points” [74]. In practice what this means is that solutions are ranked based on ‘layers’ of non-domination (as Goldberg proposed). All solutions of the same rank are initially given an identical fitness (allowing equal chance of selection amongst them). So for example, solutions of rank 1 are initially assigned a large dummy fitness value; ‘sharing’ is then used to penalise members in more crowded regions, that is, an

³Personnel responsible for final selection of the optimised solution to be used.

individual’s fitness value is divided by the number of neighbouring solutions found within a specified niche area. Non-dominated members of rank 1 are then ignored and a sub-population of rank 2 identified. Their initial dummy fitness will be less than the minimum value assigned during initialisation of rank 1s. This process continues until the entire population has been assigned a tailored fitness value.

Upon selection of the next generation a greater proportion of members on or near the current estimated Pareto Front will be copied (as they possess the highest fitness values) and thus selection pressure should progressively move the whole population forward towards the true front, whilst permitting a suitable diversity for avoidance of local optima. The main disadvantage of such an approach is the computational overhead associated with repetition of Pareto ranking and fitness assignment. More recent algorithms can achieve the same goal far more efficiently [15], they do so by simply maintaining an elitist set of parametrisations representing the current best estimate of the Pareto Front.

A more efficient approach to population niching was proposed by Horn *et al* in their Niche Pareto Genetic Algorithm (NPGA) [53]. They used a form of tournament selection where by two members are selected at random from the population as a whole and compared to a subset of the remaining populace. If one of the two members is found to be non-dominated in respect to the select subset and the other dominated, the non-dominated member is taken as the ‘winner’. In cases where both members are either non-dominated or dominated, the solution with the smallest neighbourhood is designated as that which will procreate in the next generation. This method maintained the benefits of niching whilst avoiding the computational overhead of population ranking, however, use of tournament selection requires an effective tournament size to be specified. It should also be noted that MOGA, NSGA and NPGA are all sensitive to correct regulation of their fitness ‘sharing’ factor (the size of a solution’s neighbourhood; the niche radius).

Post 1999, researchers moved towards ‘elitism’ as a standard mechanism [15], as it has been shown to enhance convergence in MOEAs and prevent the loss of good solutions once they have been found [81, 64]. Elitism refers to the process of retaining a percentage of the ‘best’ solutions from one generation to another. The concept attempts to reduce the amount of time an EA spends re-discovering previously good solutions, lost between generations. Dominated individuals may still participate in selection in population based algorithms, however, the current ‘best’ non-dominating solutions also explicitly contribute to the generation of increasingly fit members. Zitzler and Thiele’s Strength Pareto Evolutionary Algorithm (SPEA) [83] was one of the first to implement such a scheme. SPEA maintains two populations, one containing non-dominating and dominated members in the usual manner, P , and a second containing only elitist non-dominated members, P' . At each iteration non-dominated members of P are copied to P' and any member of P' subsequently found to be dominated is removed. Should P' become too large a clustering method is employed to prune solutions from only the most crowded regions of the Pareto set. This ultimately permits a nicely distributed Pareto set, spread evenly across the estimated Pareto Front.

Fitness assignment to members of P' is calculated by ranking; the ‘strength’ (fitness)

is proportional to the number of solutions each member of P' dominates in P (similar to MOGA). Fitness assignment to members of P is then calculated by “summing the strengths of all external non-dominated solutions” [83] that dominate each member of P , those with the *lowest* value being considered most fit. Individuals are selected via tournament selection (with replacement) from the union of both P and P' . Zitzler and Thiele maintain: “This mechanism intuitively reflects the idea of preferring individuals near the Pareto-optimal front and distributing them at the same time along the tradeoff surface” [83]. That is, the notion of Pareto dominance is used to distribute members of the Pareto set, rather than any explicit niching radius. However, this means that the size of P' can have a direct impact on selection pressure; if P' becomes too large, pressure to move the non-dominated front forward will drop, hence Zitzler and Thiele’s use of cluster-based pruning.

In SPEA2 [82] Zitzler *et al.* attempt to improve their first algorithm by addressing the following shortcomings in fitness assignment, density estimation and ‘archive truncation’: In SPEA solutions dominated by the same members of P' will possess an identical fitness, if P' contains only one member this can lead to a situation where all individuals in P are all assigned the same rank. With such low selection pressure, SPEA behaves like a random search algorithm [82]. Fitness calculation in SPEA2 takes into account the number of individuals that dominate a solution *and* the number of members that the solution itself dominates, using both archives P and P' . To further prevent individuals with identical fitness values, discrimination based on nearest-neighbour density estimation is applied. In SPEA2 density estimation is extended to include information from the general population in P , not just the non-dominated members of P' . The original clustering method from SPEA is also amended so that outer solutions of the estimated Pareto set in P' are not unnecessarily pruned. In trials SPEA2 was found to outperform the original implementation [82].

Knowles and Corne sought to create “the simplest possible non-trivial algorithm capable of generating diverse solutions in the Pareto optimal set”, named the Pareto Archived Evolutionary Strategy (PAES) [61]. PAES implements a simple (1 + 1)-evolutionary strategy, where a single parent solution generates a single new child. Each new child solution is compared against an elitist archive of non-dominated solutions in order to determine acceptance, it is this process that generates the selective pressure to drive the search forward. Child solutions that are not dominated by their parent or any current archive members will always be archived and ‘accepted’. That is, they will be added to the archive and replace their parent as the ‘current’ solution. Child solutions that are dominated by any current member will be rejected and will not enter the archive. In PAES the non-dominated archive has a predefined maximum size. Thus, to determine whether non-dominated child solutions should be accepted and/or archived a grid system, in objective space, is used to determine a crowding factor. If the archive is full, a non-dominated solution will only be added if it can replace an existing solution that has a higher grid-location count. This ensures that sparser areas of the estimated Pareto front are filled out and promotes an even distribution of members. The grid count is also used to determine acceptance of a child as the new ‘current’ solution in circumstances where it does not dominate the parent solution or any member of the elite

archive; out of the two, the solution with the lowest grid count becomes the new current solution (parent).

In [62] experiments with alternative forms of PAES are presented. Namely, $(1 + \lambda)$ and $(\mu + \lambda)$ variants, where μ parents are considered and λ children are generated from the current parent(s). PAES is shown to perform well in comparison to NSGA and NPGA. In chapter 6, we investigate an implementation of a similar scheme for optimisation of STCA.

An updated version of NSGA, NSGA-II was subsequently proposed by Deb *et al* in [23]. The algorithm sought to address three main criticisms of the original implementation; the large computational overhead, the lack of elitism and the necessity to supply a suitable neighbourhood (niche) sharing factor. Firstly, ranking efficiency was improved by modifying the method used to locate progressively non-dominated layers of solutions: An archive, P , is assigned all members of the current population. A member of P is copied to a separate temporary archive P' . Remaining members of P are then compared one-by-one to members of P' ; any entries of P' found to be dominated by a newly added solution are removed, similarly if the new solution is found to be dominated by any member of P' it will not be added. At the end of this procedure P' will contain a set of non-dominated solutions corresponding to rank 1. Members of P' are then copied to a new archive, F_1 , and P' is assigned an empty set. P is assigned $P \setminus F_1$, that is, members of F_1 are removed from P . The procedure now repeats, however, this time the second layer of non-dominated solutions corresponding to rank 2 will be identified and assigned to a new archive, F_2 , and so on until all non-dominated fronts have been ranked. Deb *et al* specify a complexity of $O(MN^2)$ for this new method of rank assignment (where N is the population size and M is the number of objectives), the original NSGA implementation performed the same task in $O(MN^3)$ time.

In NSGA-II elitism is implemented by combining all child solutions from one generation with the previous generation's parent population prior to performing ranking (a $\mu + \lambda$ evolutionary scheme). The concept of a user specified niche size is replaced with an automated density calculation; *crowding distance*. Crowding distance is estimated by averaging the distance (in objective space) to the two nearest solutions either side of a given member.

It would appear that NSGA-II tends to find a better spread of solutions than SPEA however, which algorithm converges better is somewhat problem dependent [23].

Hybrid algorithms

Whilst evolutionary algorithms may show greater tolerance to local optima, they do so at the expense of speed when compared to local search methods. Hybrid algorithms attempt to combine the advantages of global and local search methods together, improving speed whilst still arriving at a global optimum. Local gradient-based methods traverse the search landscape relatively quickly but can become stuck in local optima, whereas global algorithms such as EAs are far more tolerant of local optima but may 'waste' evaluations as they inherently permit exploration of less fruitful regions of search space in the hope of discovering new basins of attraction.

A distinction can be made between two different categories of hybrid algorithm, those that

are analogous to the Lamarckian theory of evolution and those that exploit the Baldwin effect [79, 49, 29]. In the Lamarckian theory, parents are capable of passing their lifetime experiences / learned behaviours on to offspring through inheritance; although not observed in nature, this mechanism can be exploited in computer simulation through modifications to genotype. When applied to the context of hybrid algorithms, modifications made to parent solutions by the local search method are used directly as the starting point for the next generation of global search candidates. That is, the local search method is used as a refinement process to members of the current global search population [29].

In contrast, the Baldwin effect describes how learned behaviours may help survival but do not directly affect genotype. That is, the benefit of one generation's experience is implicitly inherited, individuals quick to learn a beneficial behaviour survive longer and over many generations these traits may become more pronounced or instinctive. Although not directly encoded in their offspring's genotype, the behaviour of a parent generation can affect the subsequent evolution of children. The distinction in the context of hybrid algorithms is that the local search method is simply used to calculate the *fitness* of members of the global search population, that is, their genotype remains unchanged [79]. In effect, the local search is used to gauge the potential for improvement of each global search solution [29].

In either case, a hybrid algorithm typically performs a number of global search iterations, followed by refinement of population members using a local search method, before global search resumes. It is reported in literature that the use of Lamarckian and Baldwinian strategies can be more efficient and effective than a traditional GA alone [79, 49, 29]. However, a key issue that arises is how much time should be dedicated to the global search and how much to the local. Too long spent performing local search could result in evaluations being wasted exploiting a local optimum. Conversely, too many global search iterations could waste evaluations that would have been better spent refining a current member of the population. A balance must be sought between exploration and exploitation [29], the optimal configuration of which may well be problem specific.

Typically, non-adaptive hybrid algorithms are governed by three parameters; local search frequency, local search selection probability and local search duration. The frequency of local search determines how often the algorithm switches from global search to performing local search on the current population members. As the local search process can be computationally expensive, often only a proportion of the global search population is selected for refinement. This is governed by the local search selection probability. The duration of local search defines at what point the algorithm should switch back to performing global optimisation [29, 30].

Often these threshold values must be hand-crafted, requiring prior knowledge of a given domain or experimentation [30]. Additionally, the use of such constants governing the switching process may not be well suited, as it is likely that the optimal configuration will vary throughout optimisation; global search being most beneficial at the start of the process, whilst local search refinement is more favourable as the optimum is approached. One alternative to the explicit provision of 'tuned' constants is to apply an adaptive scheme.

Adaptive and self-adaptive schemes

The question ‘to what value should optimisation meta-parameters be set’ applies to all the algorithms discussed thus far. Optimal values are often problem specific and must be derived through experimentation or prior insight into the problem structure, furthermore they may vary throughout optimisation. For example, mutation rate influences the ratio of exploration to exploitation in an EA, exploration being preferred in early iterations.

The moniker ‘Evolutionary Strategy’ (ES) is perhaps the most commonly associated with adaptive mutation schemes [11, 4]. Algorithms such as simulated annealing [58, 72, 73], which are analogous to the regulated cooling of metals such that a defect free crystalline structure is formed, adapt parameters controlling the width of their mutation distribution. By slowly reducing the distribution width at each time step the frequency of smaller solution perturbations increases, permitting the algorithm to enter an exploitation phase as the optimum is approached. The time taken to converge depends on how rugged the search landscape is. It can be proven that simulated annealing will converge to the global optimum provided that the cooling schedule is sufficiently slow [44]. That is, the probability of reaching the global optimum approaches 1.0 as the cooling schedule is extended. Thus, the time required to locate the vicinity of optimum with any degree of confidence may well make simulated annealing impractical for some problem domains, an exhaustive search might even prove faster.

However, this still raises the issue as to at what rate the cooling parameter should decrease. The ideal of a problem-independent algorithm, for which no prior knowledge is assumed, cannot be fully realised if tuning of parameters that govern the optimisation process is required. If feasible, it is preferable that an algorithm self-adapt, reducing the number of operational constants to as few as possible. Addressing this issue permits the governing variables to be adjusted *throughout* optimisation, allowing the ‘optimal’ values to vary at each iteration in response to a changing comprehension of the fitness landscape. Similarly, self-adaptation may permit better assimilation into dynamic problem domains where the optimum at one time step may differ at another as the system progresses.

The method by which meta-parameters are updated may be absolute or empirical [2]. In the case of absolute update rules, a statistic, computed over several generations, is used to determine when/how modification takes place. Empirical update rules essentially extend those parameters considered for perturbation to include the meta-parameters themselves, that is, the EA *self-adapts* the adaptive meta-parameters.

Adaptive algorithms may broadly be separated into two categories; those that act at the population level and those that act on individuals [2]. The former addressing parameters global to the entire population, for instance an overall mutation rate, whilst the latter assigns adaptable meta-parameter vectors to each member of the population, these may themselves be perturbed through standard EA modifiers.

Angeline [2] highlights a third category of adaptive algorithm; those that manipulate solutions at the component level. A component level adaptive algorithm stipulates how individual *elements* are perturbed, independent from one another. The effect is similar to

that of individual level adaptations but the method offers, perhaps unnecessarily, finer grained control over how a solution evolves. This approach may introduce undue complexity into the algorithm, increasing optimisation time with little benefit to the final convergence rate [3]. The appropriate level of adaptation is likely to depend on the complexity and structure of the optimisation problem.

As an illustrative example, Espinoza *et al* [30] use a population level absolute update rule to determine switching between global and local search methods in their Self-Adaptive Hybrid Genetic Algorithm (SAHGA). When the relative coefficient of variation in population fitness between one generation and another drops, local search is invoked to provide more information and improve performance. In [68], Pulido and Coello Coello integrate population based self-adaptation into their Micro GA II in order to automate the control of crossover rate, population size, number of iterations and population niching mechanisms.

Expensive optimisation

In recent years there has been growing interest in developing efficient algorithms for expensive optimisation problems. Industrial problems are often associated with expensive evaluation functions in terms of cost or computational effort. As EAs are used to solve increasingly complex problems there is demand for optimisation on a budget.

Literature on this subject is relatively sparse and expensive optimisation is an area of ongoing research. To date there have been two main approaches for tackling expensive optimisation. The first aims to reduce the number of necessary evaluations by developing algorithms that are more efficient at decision space sampling, either through use of a partitioning scheme [60, 61, 62] or small-population techniques that still adequately approximate the Pareto front [63, 18, 60]. The second approach introduces an approximate function or surrogate-model, on which to perform optimisation [60, 59]. This permits intermediate evaluation of solutions based on an estimation of fitness, reducing the number of on-line evaluations required. Parallelisation of solution evaluations has also been suggested (see ‘Master-Slave’ parallel GAs [12, 13]), however, this does not necessarily address the number of on-line evaluations required, only the time taken to arrive at practical results. Parallelisation can also introduce synchronisation issues in certain GA frameworks that may lead to bottle-necks or wasted evaluations.

In [60], Knowles and Hughes describe two methods designed to tackle optimisation on a ‘budget’ of 250 evaluations. The first, Binary-MSOPS, is a decision space partitioning technique based around Hughes’ original MSOPS algorithm [55], whilst the second, ParEGO, outlines use of a surrogate-model and is discussed later in this section. Both methods make the assumption that, as only a limited number of iterations are to occur, solutions need not be discarded from the population. That is, unlike other EAs the computational overhead associated with maintaining a large population (with regards to ranking and comparison) is not an issue here and thus no information need be lost through pruning of solutions.

Binary-MSOPS [60] utilises a binary search algorithm to balance exploration/exploitation and direct selection towards sparser regions of objective space whilst preventing over-sampling

in more crowded areas. A binary tree is constructed and used to determine the largest empty region of a hypercube; for exploration a point is generated at random within this region (global search), and for exploitation a new point is generated within a given distance of a selected ‘good’ point that already exists in the most empty region (local search). In trials Binary-MSOPS was out-performed by ParEGO and shown to struggle when solution density at the Pareto front is low [60].

The term ‘Micro GA’ [18] has been used to refer to a subset of small-population genetic algorithms that possess a reinitialisation process, first implemented by Kirshnakumar [63]. These typically possess a working population of 3–5 individuals; Goldberg suggested that a population size of 3 is theoretically sufficient for a GA to converge [48] however, Kirshnakumar used 5 individuals [63] and more recent work by Coello Coello and Pulido uses a population size of 4 [18, 68].

In a Micro GA evolution is split into cycles. In one cycle several iterations of the Micro GA occur until ‘nominal convergence’ is achieved (this may be a constant number of iterations or based on some measure of genotypic divergence in the population). At the start of a new cycle a proportion of the best members of the previous cycle are copied into the next generation along with randomised individuals and the process starts over.

In [18, 68], Coello Coello and Pulido implement a Micro GA for multi-objective optimisation. They sought to improve the efficiency of convergence, both in terms of the computation cost associated to comparison/ranking and in the method used to traverse the search space, minimising ‘wasted’ exploratory or exploitative evaluations. In their implementation two ‘memories’ are maintained; an external memory that contains non-dominated members of the Pareto front and an internal memory designed to maintain diversity, that is itself split into replaceable and non-replaceable portions. At initialisation both the replaceable and non-replaceable memories are filled with randomised solutions. Throughout optimisation, the non-replaceable memory remains unchanged, providing a source of diversity. After each Micro GA cycle, two members of the replaceable portion of memory are compared for dominance against two newly generated non-dominating solutions from the Micro GA population. These two individuals are also compared to existing members of the external memory; replacement occurs in respect of dominance. An adaptive grid, similar to the one employed by PAES [61, 62], is used to maintain diversity in the external memory. In trials [68], the Micro GA did not always beat PAES or NSGA-II but remained competitive.

An alternative approach to expensive optimisation is to replace the costly objective function with a cheaper surrogate-model [57, 5, 59]. ParEGO [60, 59] uses a Gaussian processes model to estimate which solutions possess the largest potential for improvement and balances exploration/exploitation accordingly. The model is updated after every function evaluation, taking into account both the predicted improvement of a solution and the error in this prediction. However, the method by which cost is calculated for multi-objective problems relies on a weighted-aggregation function and requires *a priori* knowledge of cost limits for scaling. In trials enforcing a small number of iterations, ParEGO was generally found to outperform NSGA-II for 100–250 function evaluations [59].

Jin [57] highlights three main levels of approximation in surrogate-models; problem, functional and evolutionary. At the ‘problem’ level, the problem to be solved is itself replaced by a similar system that is easier to solve. Jin cites replacing Navier-Stokes equations with Euler equations for optimisation on fluid dynamics problems, at the expense of excluding viscosity, mass diffusion and thermal conductivity calculations. The ‘functional’ level refers to the substitution of a fitness function with a surrogate-model. For instance, a neural net (see [10, 25] and section 7.2, page 128 for more on neural nets) could be trained to approximate the response (fitness) of a system to a given set of inputs (solution parameters). Polynomial models, Kriging models and Support Vector Machines are also given as examples of approximate functions by Jin. Finally, the ‘evolutionary’ level refers to fitness inheritance in evolutionary algorithms, whereby the fitness of offspring is estimated from their parent(s). The method used in ParEGO is an example of this type of approximation.

Summary

In this sub-section we have presented a brief overview of how Evolutionary Algorithms may be applied to multi-objective optimisation. We have established how the field has developed from early composite based approaches to modern day, elitist Pareto-dominance methods. Initial composite functions required prior advanced knowledge of objective costs, tended to generate a single solution – requiring multiple runs with varied weights to generate a trade-off surface, and in the case of weighted-sum approaches, could not reach non-convex regions of a Pareto Front. In contrast, modern Pareto-dominance based methods are able to obtain a population of non-dominating candidates approximating the Pareto Front in a single run and when implementing an elitist archiving approach, efficient convergence may be observed. Whilst some recent aggregation frameworks such as MSOPS-II have been successfully applied to many objective problems, Pareto dominance based methods are very effective for problems with 2, 3 or 4 objectives, as is the case for STCA.

Although NSGA-II is often used as the benchmark against which other MOEAs are compared [15], the algorithm that we initially implement in optimisation of STCA is most similar to PAES. PAES provides a simple baseline framework at the expense of some coverage of the Pareto front when compared to algorithms such as NSGA-II [23]. We attempt to compensate for this by using a selection scheme that is biased towards sparser areas of the ROC, encouraging spread. However, the implementation is not that dissimilar to the grid method employed in PAES.

Like PAES we utilise a $(1 + 1)$ evolutionary algorithm, however, an alternative selection scheme is employed that does not enforce the *same* ‘single-parent’ to be perturbed/updated each iteration. We choose the PAES framework due to its simplicity and the ease with which it may be extended to integrate alternative schemes. The complications of rank-based fitness methods are avoided and non-dominance alone is used to determine solution superiority. We do not limit the size of the elite non-dominated solutions archive, thus no pruning scheme is required. Instead of maintaining a ‘current’ single-parent solution, each iteration a new parent is randomly selected from the elite archive set, with bias towards sparser areas of the

Pareto Front.

The use of a hybrid scheme was deemed too expensive due to the additional evaluations that would be required during local-search cycles. Conversely, algorithms for optimisation on a budget such as Binary-MSOPS and ParEGO assume that no solution is discarded. Although STCA is an expensive optimisation problem, due to the size of the decision space we expect the number of necessary iterations to be in terms of thousands rather than hundreds (indeed preliminary studies have shown this to be the case [37, 35]). This makes the retention of all solutions generated impractical, discard is necessary to prevent excessive computational overhead when managing the population. Implementation of an adaptive mutation scheme was also ruled out as preliminary work [37, 35] suggested that optimisation of STCA is relatively insensitive to mutation width. In addition, implementation of an adaptive scheme at the individual level would likely be impractical due to the size of STCA parametrisations. During the course of optimisation there would be too few perturbations made to each individual parameter to build up accurate statistics on which to base an absolute adaptation metric. Similarly, use of an empirical self-adaptive scheme would only serve to further increase the search space complexity. In the following chapter we give further detail as to how optimisation of STCA will be achieved.

2.5 Chapter summary

This chapter has introduced many of the background methods and techniques associated to air traffic control. We have seen how air space may be divided into manageable subsets, each with its own purpose and associated traffic characteristic. The Short Term Conflict Alert system has been described in the context of an imperfect classifier, designed to aid controllers by predicting potential loss of aircraft separation before such events occur.

STCA may be manually tuned, using historical datasets, to meet the requirements of a given airspace region. However, the many governing system parameters make this process complex and arduous. Maintenance personnel are required to finely balance the resulting ‘wanted’ and ‘nuisance’ alerts, reaching an acceptable trade-off.

We have seen how the concept of evolutionary algorithms might be applied to automate the optimisation of STCA-type systems. Repeated perturbation of dominant parent parametrisations should result in increasingly fit candidates over successive child generations. This iterative process can be used to automatically locate a Pareto optimal ROC curve, characterising STCA performance and presenting maintenance personnel with a range of possible configurations to be used. We stress the expensive nature of STCA optimisation and underline the necessity of keeping the number of function evaluations to a minimum. In the next chapter we describe a ‘single archive’ algorithm for optimising STCA.

Chapter 3

The single-archive optimiser

The following sections will describe the implementation of a simple evolutionary algorithm. This will optimise the trade-off between wanted alerts and nuisance alerts generated by the NATS STCA system. Results of a trial run will be compared to the NATS, manually tuned, current operating point.

This basic optimisation procedure will form the foundation architecture from which all further enhancements will be derived; in chapter 4, sections 4.1 and 4.2, we improve its efficiency by looking at methods of dividing the parameter search space into subsets.

3.1 The single-archive optimiser

The use of the term ‘single-archive’ is used here to differentiate the optimiser model from later enhancements. It can be thought of as a more traditional Multi-Objective Evolutionary Algorithm (MOEA) where all members of the population are kept in one ‘archive’ or elite set. Elements of this set are mutually non-dominating and form the current approximation to the Pareto front.

Parent members are selected at random from the archive, A , and perturbed in some way to create a child parametrisation or ‘solution’. The child parametrisation is then sent for evaluation on the off-line STCA model (CAMPAP). We use an ‘elitist’ algorithm whereby the evaluated child is only added to the archive of parents if it is not dominated by the existing members. Any existing members that the *child* dominates are discarded from the archive as the child is added. Therefore the archive, A , is maintained as a mutually non-dominating set (approximating the Pareto front) that can only move towards the Pareto front.

Figure 3.1 outlines the main loop of the algorithm. Starting at state 1, an existing member of archive A is selected as a ‘parent’ parametrisation to be perturbed. The parent parametrisation is modified in some way, forming a new ‘child’ parametrisation that is distinct from the original parent (state 2). The child parametrisation is evaluated on the off-line STCA model and associated true positive and false positive rates determined. State 3; if the child parametrisation is not dominated by any existing member of archive A , it is added to A and any members that it dominates are discarded. State 4; alternatively if the child

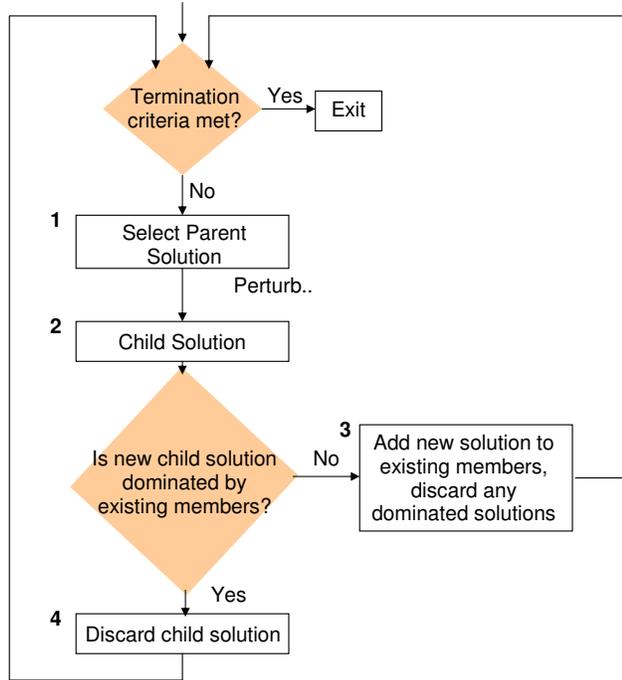


Figure 3.1: Flowchart of a basic evolutionary algorithm.

parametrisation *is* dominated by any existing member of archive A , it is discarded. The process is then repeated from state 1.

Successive iterations of this basic algorithm will ensure that the elite set of parametrisations can only move towards the Pareto front (i.e. accepted perturbed parametrisations can never result in a worse approximation to the front), although it may converge to a local optimum rather than the global Pareto front. See appendix B for the optimiser’s data flow diagram; details of the STCA wrapper may be seen in appendix C.

Algorithm 1 outlines the procedure more formally. At line 1 the archive, A , is initialised (see section 3.1.1 for initialisation options). An initial set of parametrisations is evaluated on the STCA system and added to the archive depending on dominance (see algorithm 2, *update*). At line 3 a parent parametrisation, θ , is selected from archive A (see section 3.1.2 for selection schemes). The parent parametrisation is then perturbed to create a new child parametrisation, θ' , at line 4 (see section 3.1.3 for more on perturbation). At line 5, the child parametrisation, θ' , is evaluated on the STCA system, its associated true and false positive rates are obtained. The child parametrisation is then updated to archive, A at line 6. The algorithm then loops back to line 2 and repeats the process N times.

Algorithm 2 outlines the *update* method used to determine whether the new parametrisation, θ' , should be added to archive A . If the new parametrisation is found to be non-dominated (line 1) in respect to existing members of archive A , then dominated members of A are removed (line 2) and the new parametrisation, θ' , is added to A (line 3). Otherwise the new parametrisation is discarded (not added).

Algorithm 1 Single-archive Main()

Require: N Number of iterations

1: $A := \text{initialise}()$ Initialise archive A (section 3.1.1)
2: **for** $n := 1$ to N **do**
3: $\theta := \text{select}(A)$ Select a parent, θ , to perturb (section 3.1.2 and algorithm 3)
4: $\theta' := \text{perturb}(\theta)$ Perturb parent to create a new child, θ' (section 3.1.3)
5: $(TP(\theta'), FP(\theta')) := STCA(\theta')$ Evaluate child's TP and FP rates on STCA
6: $\text{update}(A, \theta')$ Update archive, A , with child θ' (algorithm 2)
7: **end for** Loop for N generations

Algorithm 2 $\text{update}(A, \theta')$

1: **if** $\nexists(\phi \in A \text{ s.t. } \phi \preceq \theta')$ **then**
2: $A := \{\phi \in A \mid \theta' \not\preceq \phi\}$ Remove dominated members of A
3: $A := A \cup \{\theta'\}$ Add child θ' to A
4: **return** *True* Return true, indicating that the child, θ' , was added
5: **end if** Else discard the child, θ'
6: **return** *False* Return false, indicating that the child, θ' , was not added

3.1.1 Initialisation of the archive

The optimiser must begin with a non-empty set of parametrisations from which to select a ‘parent’; the ‘initial set’. Five options for generating the initial set exist, any combination of which may be selected.

Initialisation options:

1. Initialise with default values parameter set. Default parameter values may be defined in the XML ‘parameter-map’. In our trials, these were set to a ‘conservative’ value selected by analysing all previously used STCA settings (over the previous 12 years) for each parameter. What we mean by this is that initialisation and perturbation (see section 3.1.3) of parameters is restricted to ranges previously used by NATS at the Current Operating Point (COP - parameter configuration used on the current live STCA system). ‘Conservative’ values were determined by analysing all previously used COP STCA settings (over the past 12 years) for each parameter and its associated regions (if applicable). This was done as a validation exercise, providing an assurance that the optimised system was still operating within the usual parameter ranges and also ensuring that the optimiser could reach (and hopefully exceed) the current NATS COP without explicit knowledge of it. See Chapter 5 for a discussion of using values outside of the conservative range.
2. Initialise with ceiling values parameter set: Conservative ceiling parameter values may be defined in the XML parameter-map. These were set to the maximum value found to have previously been used, over the past 12 years, for each parameter.
3. Initialise with floor values parameter set: Conservative floor parameter values may be

defined in the XML parameter-map. These were set to the minimum value found to have previously been used, over the past 12 years, for each parameter.

4. Initialise with randomised values parameter set(s): Parameter sets may be generated by randomly setting parameter values uniformly selected from the conservative maximum – minimum range defined in the parameter map. The number of sets to be created is configurable; however, in the work reported here it was fixed to 100.
5. Initialise with restored parameter set(s): This option is included so that optimisation runs may be resumed after pausing/stopping or have customised parameter sets submitted. Any STCA parameter files found in a designated directory will be loaded into the optimiser.

It should be noted that, as initialised solutions are added to the archive, dominance is adhered to. That is, only non-dominated members are retained, those found to be dominated are still discarded in this initialisation phase.

3.1.2 Parent selection

A method is needed to identify parents to select for perturbation to produce the next (hopefully dominating) generation of child parametrisation. A good selection scheme should promote an even distribution of parametrisations across the Pareto front as well as aid convergence.

Two modes of parent selection are available to the ‘single-archive’ optimiser; ‘Uniform’ and ‘Uniselect’. ‘Uniform’ selection simply allows all parent members to be selected with an equal chance. However, this may lead to clustering of parametrisations as sections of the Pareto front with a higher density are more likely to have parametrisations selected for perturbation than sparser areas. As child parametrisations are likely to be of similar fitness to their parent they may join them at the front accentuating the clustering.

Uniselect ([73], see also [40] for similar PQRS method) allows a bias to be given towards selecting those members in a more sparsely populated region of the ROC. This encourages the ‘filling-out’ or spreading of points across the curve. In addition, the scheme can be extended to allow a user to explicitly specify the probability of just the end points of the archive curve being selected, the intention being that in forcing the ends to be picked a certain percentage of the time we coerce the curve to spread/grow outwards quicker.

Uniselect selects a random number from the current range of values associated to one objective and then computes which existing parametrisation is closest to this point, using it as the selected parent member which goes on to be perturbed. Each time a parent is selected the objective is switched between TP and FP.

Algorithm 3 $\text{uniselect}(A)$

Require: n	The current iteration number
Require: D	Number of objectives (dimensions) e.g. TP and FP
Require: $pMin$	Probability of selecting min end point
Require: $pMax$	Probability of selecting max end point
1: $p := \text{rand}()$	The probability of selecting an end point
2: $i := n \text{ modulo } D$	Objective on which to perform Uniselect
3: if $p \leq pMin$ then	
4: $v := 0$	Results in selection of the min end point
5: else if $p \leq (pMin + pMax)$ then	
6: $v := 1$	Results in selection of the max end point
7: else	
8: $v := \text{rand}()$	Results in u being drawn uniformly
9: end if	
10: $u := v \times (\max_{\theta \in A} f_i(\theta) - \min_{\theta \in A} f_i(\theta)) + \min_{\theta \in A} f_i(\theta)$	Scale u to the current objective's range
11: return $\arg \min_{\theta \in A} f_i(\theta) - u $	Select and return the closest element, θ , to u as parent member

Uniselect is defined as follows [73]; where i is the currently chosen objective, each time a parametrisation is to be selected from an archive A , a random number u is drawn uniformly in the range:

$$[\min_{\theta \in A}(f_i(\theta)), \max_{\theta \in A}(f_i(\theta))] \quad (3.1)$$

and the element of A to be perturbed is the element with i^{th} coordinate closest to u :

$$\theta = \arg \min_{\theta \in A} |f_i(\theta) - u| \quad (3.2)$$

Algorithm 3 outlines the *uniselect* method with end point selection. Line 1 determines the probability, p , of end point selection and line 2 determines on which objective, i , selection should be based for the given iteration, n . At line 3 if the probability of end point selection is equal to or less than $pMin$ then, at line 10 u will equal the minimum value belonging to the current objective's range. If p is greater than $pMin$ but equal to or less than $pMin + pMax$ (line 5) then, at line 10 u will equal the maximum value belonging to the current objective's range. Otherwise, at line 10 u will be uniformly drawn from a value *within* the current objective's range. At line 11 the parametrisation with objective i closest to u is selected from archive A and returned as parent θ .

For the purpose of our trials, the probability of end point selection was set to 0.1 for both $pMin$ and $pMax$ (leaving a 0.8 probability of Uniselect selection across all parent members).

3.1.3 Perturbation

Selected parent parametrisations are perturbed (modified in some way) to create a new ‘child’ parametrisation. The probability of a parameter value being selected for perturbation is configurable, however, for our purpose was set to 0.1, i.e. in one perturbation of a parametrisation on average 10% of the parameters will be modified¹. Multiple regional values defined for each named parameter are essentially treated as separate parameters for the purposes of perturbation.

Rules as to the type of modifications that are permitted may apply. For instance, a specified parameter within the parametrisation may be discrete valued or continuous and, if continuous, is likely to have upper and lower bounds associated.

Perturbation is performed based on the parameter ‘type’. Discrete valued or categorical parameters, which in STCA systems have only a few possible categories, are modified in a purely uniform manner, that is, the parameter has an equal chance of being amended to any of the other possible values that have been defined. For example, the parameter ‘SCTHRESHOLD’ may only be set to 1, 2 or 3; if it is currently set to 2 upon perturbation it will be amended to either a 1 or 3 (there is equal probability of either value).

Continuous-valued parameters (those that may be modified over a defined range) can be perturbed based on Gaussian or Laplacian distributions (see figure 3.2). Use of the heavy-tailed Laplacian distribution is designed to promote exploration: while there is a greater probability of small increments/decrements being made to the parameter value, occasional larger ‘jumps’ are more likely, aiding exploration [80].

Perturbation of a parameter θ_k

$$\theta'_k := \theta_k + \epsilon \tag{3.3}$$

where ϵ is drawn from a Gaussian or Laplacian density.

Gaussian:

$$p(\epsilon) \propto \exp \left\{ -\frac{\epsilon^2}{2\sigma^2} \right\} \tag{3.4}$$

Laplacian:

$$p(\epsilon) \propto \exp \left\{ -\frac{|\epsilon|}{\sigma} \right\} \tag{3.5}$$

where σ sets the width of the density. If θ'_k falls outside the permitted parameter range, then ϵ is resampled until it is within range.

¹This value was determined through prior experimentation [37, 35].

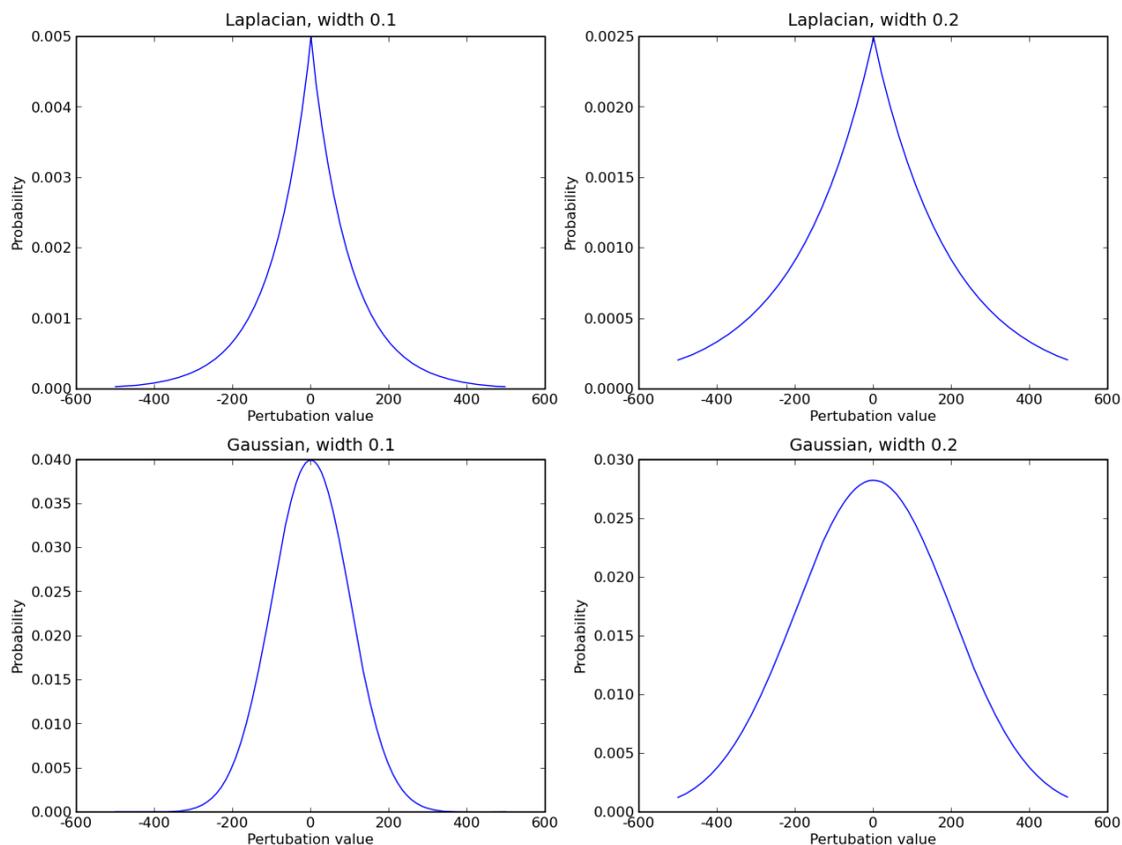


Figure 3.2: Laplacian & Gaussian perturbation distributions at widths of 0.1 and 0.2 (parameter range -500 to 500).

Figure 3.2 shows a parameter whose values are perturbable across the range -500 to 500. The scale variable ‘width’ defines what percentage of the range the distribution should be based on. Increasing the width flattens the distribution resulting in a higher probability of larger perturbations. If the new perturbed value falls outside of the parameter’s permitted range then the perturbation is resampled. During trials, the width was set to 0.1 (10% of a parameter’s range). Prior work [37, 35] suggested that optimisation of STCA is relatively insensitive to mutation width and early trials determined that a value of 0.1 was appropriate.

The definition of parameter names, data-types, precision, regions and possible categories/ranges are contained in an XML ‘parameter-map’.

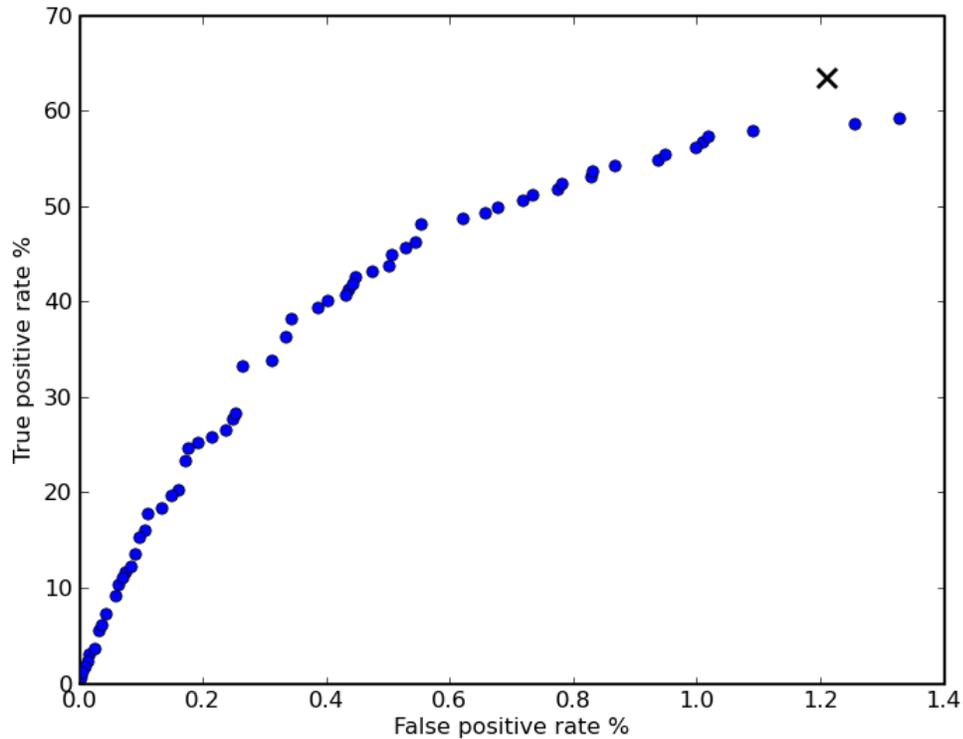


Figure 3.3: Single-archive optimisation results, MACC dataset (13722 iterations). The blue circles represent parametrizations located by the optimiser, the cross marks the NATS COP.

3.1.4 Results of the single-archive optimiser

Figure 3.3 shows the archive contents after 13722 iterations on two weeks ($\approx 70,000$ aircraft pairs) MACC July 2007 data. After initialisation from 100 randomly generated parametrizations, the optimiser has located an ROC curve consisting of 63 parametrizations ranging from 0.62% to 59.26% True Positive (TP) and 0.00% to 1.33% False Positive (FP). Note the convexity of the curve, which possesses no predominant discontinuities or concavities. There is good coverage over the breadth of the arc however, clustering is observed towards lower TP/FP rates and the density of solutions is seen to reduce towards higher TP/FP reaches. This distribution is likely due to class-skew present in the dataset, the comparatively few positive instances leading to increasingly sparse solutions as TP rate increases.

Due to time constraints, in order to reduce the time taken to evaluate each parametrization we used half the number of aircraft pairs that would normally be used when manually optimising. In addition, this optimisation was considered conservative in the sense that we limited the ranges over which the optimiser was allowed to perturb values. This was done to ensure that the optimised system was still operating within ‘normal’/stable ranges.

It should be noted that the NATS manually tuned Current Operating Point (COP) was not included in the initial set. This was done to test the ability of the optimiser to better the COP without supplying it. Instead the initial set consisted of 100 parametrizations, each

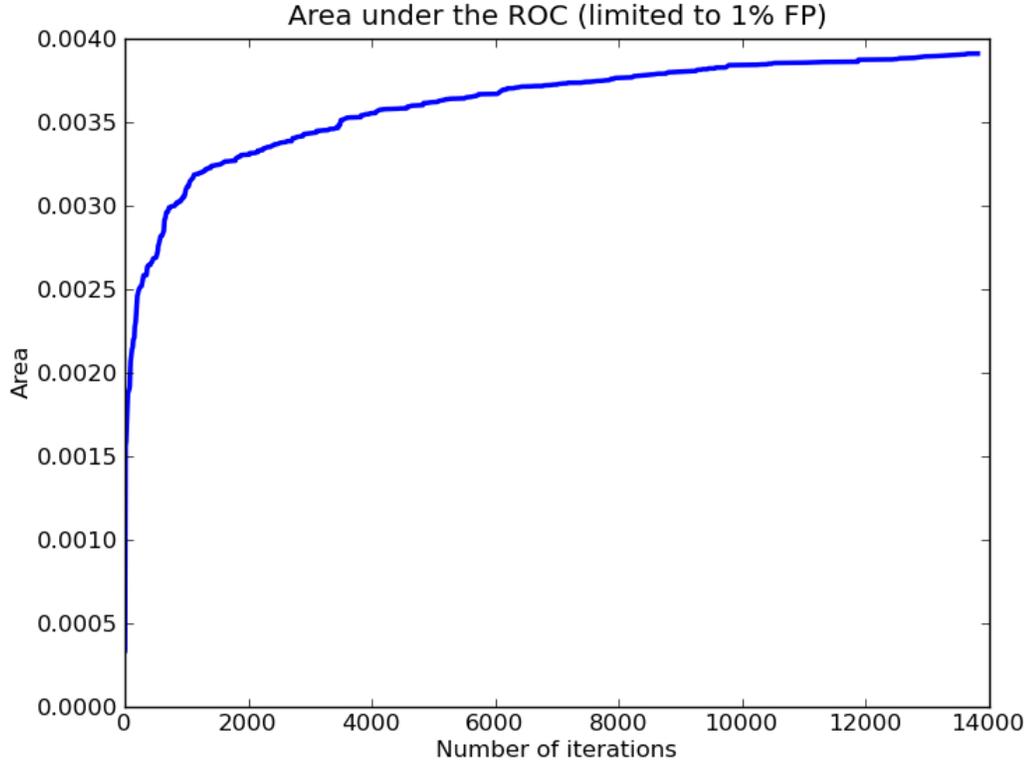


Figure 3.4: Single-archive area under ROC, MACC dataset. The graph shows the growth with iterations of the area under the ROC for false positive rates up to 1%, as this is the region of the curve relevant for NATS operation.

initialised with random parameter values. The COP for this dataset was 63.58% TP and 1.21% FP (highlighted as the black cross in figure 3.3). Thus, on this occasion the optimiser did not generate an ROC with any points that dominate the COP.

The area under the ROC is often used as an indicator of the quality of a classifier as it represents the ability of a classifier to separate the data [36]. Referring to figure 3.4, there is an initial rapid increase in the area under the ROC, as the optimiser makes larger jumps towards more optimal parametrisations, followed by a gradual levelling as a ‘fine-tuning’ phase is entered. We can see that, although the area under the ROC is beginning to level, it is still increasing. Thus, one explanation could be that the optimiser did not achieve the required number of iterations to push the front past the manually tuned operating point.

In practise the COP or archive contents from a previous optimisation run would be used to seed subsequent trials. Figure 3.5 demonstrates how including the COP in the initial set aids convergence of the Pareto front. After just 100 iterations (figure 3.5(a)) the estimated Pareto front initialised from the COP (red squares) is much further ahead than that generated from 100 random parametrisations. However, as is to be expected, the parametrisations located cluster around the COP. The random initialisation has aided spread of the ROC into higher and lower TP and FP rates. By 2600 iterations (figure 3.5(d)) the randomly initialised trial has begun to catch up to the COP initialised trial and in fact intersects the archive at $\approx 80\%$

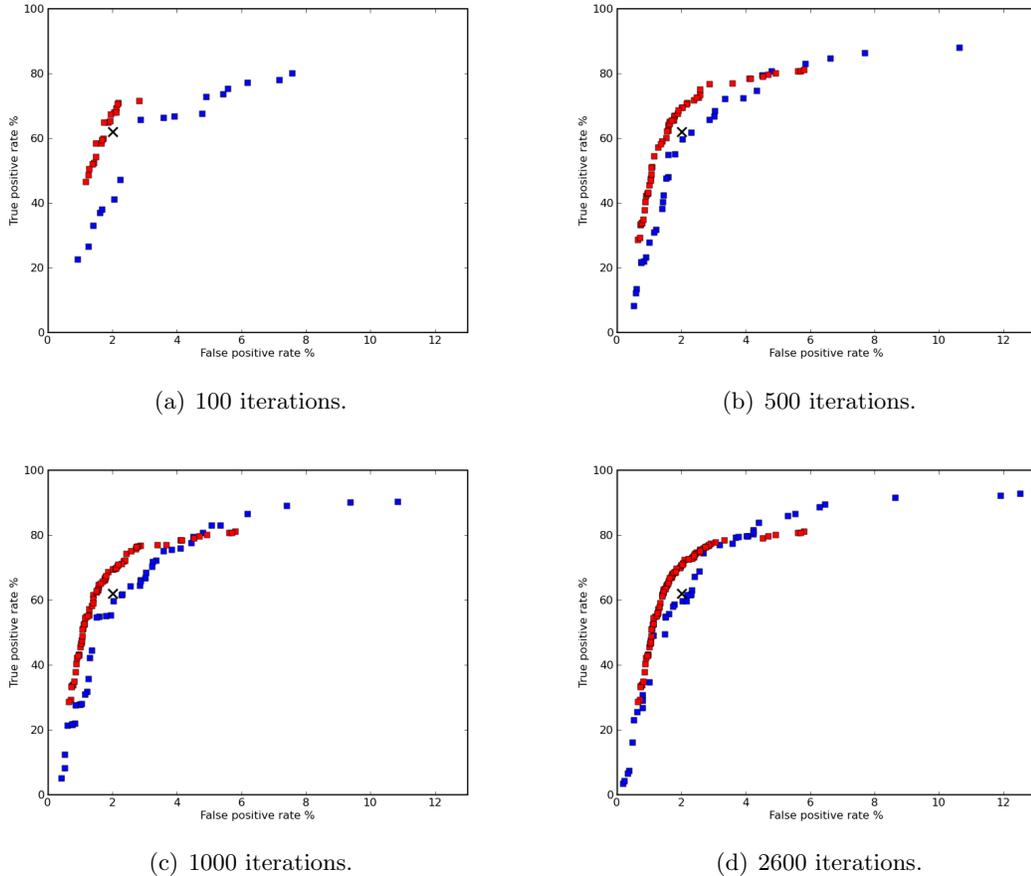


Figure 3.5: Illustration of Single-Archive convergence from different initialisations on the LTCC ESTCA dataset. Initialisation from COP (red squares) vs. initialisation from 100 random parametrisations (blue squares). The COP is marked as a black cross.

TP, though some solutions at the ‘knee’ of the curve do not yet coincide and still lie behind. Although the COP trial’s archive has started to spread into higher and lower TP and FP rates, it does not yet match extremities of the randomly initialised trial. We can surmise that including the COP in the initial set can help to speed up convergence however, the presence of randomised parametrisations can aid spread.

It should be noted that the trials shown in figure 3.5 were performed on an LTCC ESTCA dataset using the more advanced optimisation algorithm from chapter 5, thus they are not directly comparable to figure 3.3. They are simply included as a demonstration of the effect of including the COP in the initial set.

3.1.5 Optimisation speed

The principal limiting factor in the optimisation process is by far the time taken to evaluate each parametrisation. However, the efficiency with which STCA processes datasets cannot be improved (even if possible), as it is important for the safety case that the STCA code-base remains unaltered. Ideally, we would like to maximise optimisation speed, giving better

results in an equivalent time span.

Parallelisation of parametrisation evaluations is perhaps the most obvious way to increase throughput. With this in mind, the optimiser is capable of ‘driving’ multiple CAMPAP instantiations. The current configuration uses 3 servers² allowing an evaluation to be submitted to each of the Alpha box’s three CPUs. However, this approach presents a new problem; a new parametrisation, θ' , may be selected for perturbation and submitted for evaluation before the results of the of a previous parametrisation’s, ϕ' , evaluation are known. Therefore the new parametrisation, θ' , may already be dominated by ϕ' before it even finishes evaluation. The evaluation of θ' is thus wasted. In practise however, the Pareto front is moving slowly so likelihood of clashes is relatively small and the speed up is almost linear in the number of CPUs available.

As the Alpha system is a multi-user environment it is unacceptable to monopolise all available resource. To assess the likely impact of the optimiser on other users we monitored the overall system load over a period of 10 weeks. This allowed us to design a scheduler into the optimiser that enables or disables the use of each evaluation-server at different times of day and can start or stop the optimiser automatically. See appendix D for the full resource investigation report.

By the same sentiment, we highlight that it is not feasible to conduct multiple trails with random seeds, nor fully investigate the effect of varying meta-parameter configurations on a system of this size, as doing so would limit our ability to explore alternative optimisation schemes. Clearly, there is a compromise to be made as we are limited by both time and computational resource.

3.2 Summary

This chapter has presented how a set of basic initialisation, selection, perturbation and archive update functions may be defined for a multi-objective evolutionary algorithm. Results of trialling the algorithm in its present form have shown that even after 13722 optimisation iterations, the resulting ROC has not managed to locate or dominate the NATS current operating point. However, it is expected that trials would normally be initialised using the COP or archive contents from a previous run, this would help to speed up convergence. In addition, regardless of how long the optimisation process takes, there is benefit to be found in that time taken is automated CPU time *not* manual tuning time.

The speed of parametrisation evaluations on the off-line STCA model has been highlighted as a major limiting factor. While parallelisation has been suggested as a possible solution, this method seeks to improve on speed by simply increasing the number of evaluations possible in a given period (tapping extra processing power). In the next chapter the ‘multi-archive’ optimiser will be introduced. This seeks to improve optimisation speed by *reducing* the number of evaluations that are *necessary* to generate an ROC curve of equal progression.

²Each optimiser server ‘wraps’ system calls and I/O to and from CAMPAP.

Chapter 4

A multi-archive approach to optimisation

In this chapter we examine how the problem may be divided into multiple sub-optimisations, each with its own archive, with the aim of promoting faster convergence. Section 4.1 describes how the single-archive optimiser may be extended to perform multiple optimisations in parallel on subsets of an STCA parametrisation. Based on STCA airspace regions we divide the parameter space into almost independent segments and thus hope to significantly reduce the number of iterations required to generate an optimal ROC curve. The search space should be traversed in a more efficient manner, allowing a greater number of parameter values to be ‘intelligently’ perturbed between evaluations on the off-line STCA model. The results of a trial run are compared to the previous single-archive optimisation and the NATS, manually tuned, current operating point.

In section 4.2 we describe how the concept of multiple archives may be further extended. A method of recombining evaluated parameter subsets, to produce new ‘estimated’ parametrisations, is presented. The results are compared to previous single and multi-archive mode trials.

Finally, in section 4.3 the efficacy of using multiple archives is discussed. Dependencies between regional parameters can introduce noise reducing the effectiveness of parametrisation perturbations. We examine the causes of dependency and suggest alleviating techniques.

4.1 The multi-archive optimiser

As previously described in section 2.3.3 (page 19), the off-line STCA model, CAMPAP, is capable of reporting STCA performance on a region by region basis. If we were to assume that each region’s corresponding parameters are independent of the other regions’ parameters, then optimisation of the regional subsets could be performed in parallel by splicing non-dominating regional subsets into one parametrisation for simultaneous evaluation on STCA.

Based on the fact that we can obtain the number of True Positive and False Positive alerts found in each region of airspace, not just the overall STCA alerts rate, we can perform separate

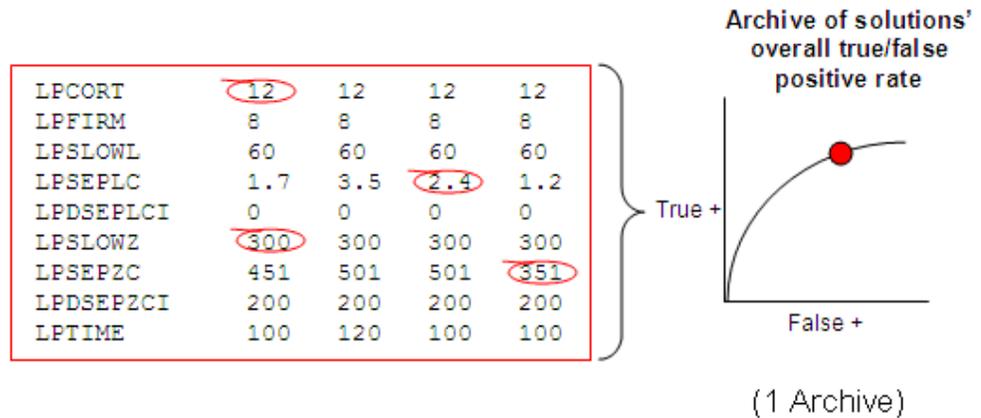


Figure 4.1: Region ‘splicing’. Parametrisations are maintained in separate archives each concerned with evaluating dominance for a particular regional parameter subset and the associated TP and FP rates resulting in that region. For evaluation purposes non-dominated parameter subsets are selected from each regional archive and combined to create a new parametrisation that can be evaluated on the STCA system.

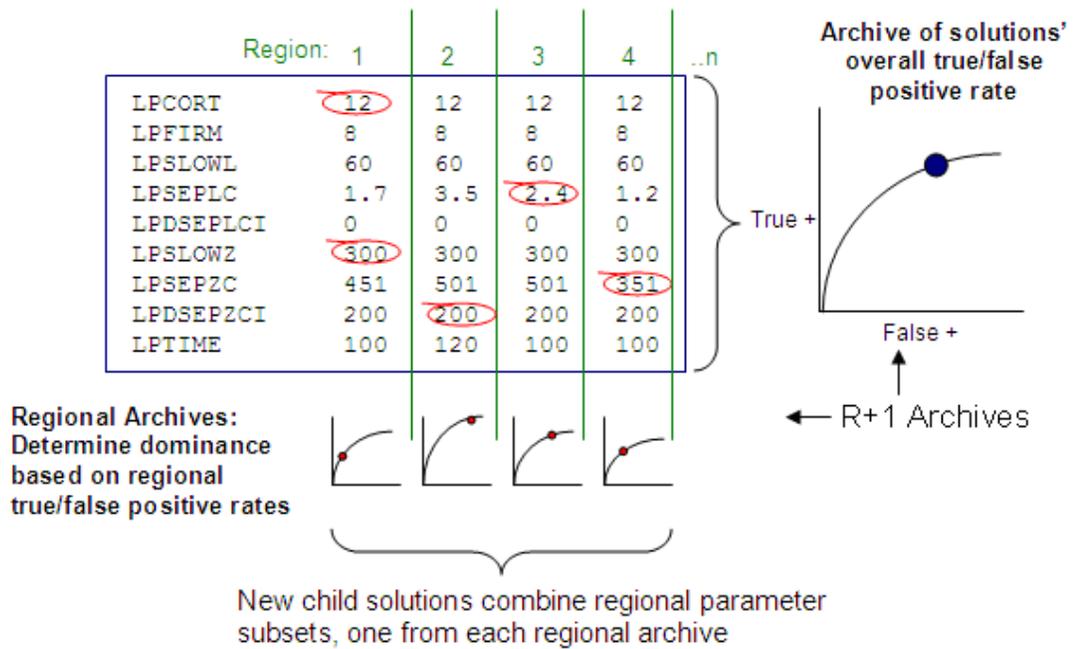
optimisations on regional subsets of the STCA parameters. Instead of treating parameters as one whole set to be optimised, the space can be divided such that $R + 1$ optimisations can take place in parallel. Where R is the number of regions in the parameter set; we perform R simultaneous optimisations on each parameter subset, as well as maintain an ‘overall’ archive that evaluates the dominance of each parametrisation as a whole. New parametrisations generated by the optimiser are a composite of the non-dominated subsets found in all the other archives (figure 4.1).

Parameter perturbation is performed on regional subsets of parameters. Perturbation occurs at least once in each region. Essentially, the search space is thought of as R separate optimisations to be performed and evaluations thus happen in parallel. This reduces the total number of evaluations that are necessary to generate the overall optimal ROC.

Referring to figure 4.2(a), we see the Single-Archive optimiser’s perturbation scheme. Here parameter perturbation is performed across the whole set, regardless of regional values. Figure 4.2(b) illustrates the new Multi-Archive scheme where by perturbation is performed based on regional subsets of parameters. The subsequent splicing of regional subsets to form a child parametrisation could be thought of as a form of crossover in genotype. Non-regional parameters are selected from the ‘overall’ archive and combined with the regional subsets to



(a) Single-archive perturbation scheme.



(b) Multi-archive perturbation scheme.

Figure 4.2: Perturbation schemes: The parameters circled in red represent those selected for perturbation in one iteration. Figure 4.2(a) represents single-archive selection, parameters are selected across all regions with equal probability. The resulting overall TP/FP rate for the given solution determines whether it enters the archive. Figure 4.2(b) illustrates multi-archive selection, the perturbation scheme enforces that at least one parameter from every region is perturbed. Regional TP/FP rates are used to determine which subsets of parameters enter their corresponding regional archive. The overall TP/FP rate is still used to determine the fitness of the solution as a whole.

complete the new parametrisation’s parameter set.

The same options for initialisation, parent selection and perturbation are available in the Multi-Archive optimiser as with the Single-Archive mode. Algorithm 4 outlines the multi-archive procedure more formally. At line 1 a set of parametrisations (I) is initialised and each member evaluated on the STCA system (see section 3.1.1 on page 47 for initialisation options). However, dominance of the members is not determined at this stage. Some of the parametrisations may therefore dominate others. Lines 2 to 5 initialise the ‘overall’ archive, A , and regional archives, $\{A_i\} i = 1, \dots, R$, as empty sets. Lines 6 to 12 update each of the archives with each parametrisation, θ , in set I . At line 7 the ‘overall’ archive, A , is updated with θ using the parametrisations overall TP and FP rates (see algorithm 2 on page 47 for details of how the dominance of parametrisations is determined during archive `update`). At line 9 a subset of parameters, ϕ , corresponding to region i , is extracted from θ . The regional parameters subset, ϕ , is then updated to the corresponding regional archive A_i at line 10. Dominance is determined based on the subset’s regional TP and FP rates, not the overall parametrisation’s rates. Each of the archives has now been initialised.

The evolutionary optimisation loop is entered at line 13. At line 14 a parent parametrisation, θ , is selected. The parent is created by combining parameter subset’s from archives $A, \{A_i\}$ using either `compositeSelect` (algorithm 5) or `compositeUniselect` (algorithm 6). The parent parametrisation is then perturbed to create a new child parametrisation, θ' , at line 15 (see section 3.1.3 on page 50 for more on perturbation). Perturbation must ensure that at least one parameter from each regional subset is changed (otherwise STCA evaluation of the parametrisation, θ' , in that region is wasted). At line 16 the child parametrisation, θ' , is evaluated on the STCA system. The overall TP and FP rates for the parametrisation are obtained, along with the associated regional TP and FP rates. At line 17 the ‘overall’ archive, A , is updated with θ' using the parametrisations overall TP and FP rates. At line 19 a subset of parameters, ϕ , corresponding to region i , is extracted from θ' . The regional parameters subset, ϕ , is then updated to the corresponding regional archive A_i at line 20. Dominance is determined based on the subset’s regional TP and FP rates, not the overall parametrisation’s rates. The algorithm then loops back to line 13 and repeats the process N times.

The `compositeSelect` method described in algorithm 5 is used to combine parameter subsets from all archives to create a new composite parametrisation. Selection of parameter subsets from the archives is based on the *uniform* and *uniselect* methods previously described in section 3.1.2 on page 48.

At line 1 a member, θ , of the ‘overall’ archive A is selected (using the uniform or uniselect methods). At line 2 the non-regional parameters subset is copied from θ and assigned to θ' . Lines 3 to 6 concern the selection and splicing of regional parameter subsets from $\{A_i\}$. At line 4 a regional parameter subset, ϕ , is selected from regional archive A_i (using the uniform or uniselect methods). The selected subset, ϕ , is appended to θ' at line 5. After all regional subsets have been appended (line 6), the composite parametrisation, θ' , is returned (line 7).

Algorithm 4 Multi-archive Main()

Require: N Number of iterations
Require: R Number of STCA parameter regions

1: $I := \text{initialise}()$ I contains a set of parametrisations, some of which may dominate others (section 3.1.1, page 47)
2: $A := \emptyset$ Initialise ‘overall’ archive as an empty set
3: **for** $i := 1$ to R **do**
4: $A_i := \emptyset$ Initialise each regional archive as an empty set
5: **end for**
6: **for** θ in I **do**
7: $\text{update}(A, \theta)$ Update ‘overall’ archive, A , with θ (algorithm 2, page 47)
8: **for** $i := 1$ to R **do**
9: $\phi := \text{extractRegion}(\theta, i)$ Extract parameter subset, ϕ , corresponding to region i from parametrisation θ
10: $\text{update}(A_i, \phi)$ Update regional archive, A_i , with ϕ (algorithm 2, page 47)
11: **end for** End update of regional archives with current member, θ , of I
12: **end for** End iteration through parametrisations in I

13: **for** $n := 1$ to N **do**
14: $\theta := \text{select}(\{A_i\}, A)$ Select a parent, θ , to perturb; **select** is either **compositeSelect** (algorithm 5) or **compositeUniselect** (algorithm 6)
15: $\theta' := \text{peturb}(\theta)$ Perturb parent to create a new child, θ' ; **peturb** must ensure that at least 1 parameter from each region is changed
16: $(TP(\theta'), FP(\theta')) := \text{STCA}(\theta')$ Evaluate child’s TP and FP rates on STCA
17: $\text{update}(A, \theta')$ Update ‘overall’ archive, A , with child parametrisation θ' (algorithm 2, page 47)
18: **for** $i := 1$ to R **do**
19: $\phi := \text{extractRegion}(\theta', i)$ Extract parameter subset, ϕ , corresponding to region i from parametrisation θ'
20: $\text{update}(A_i, \phi)$ Update regional archive, A_i , with ϕ (algorithm 2, page 47)
21: **end for** End update of regional archives with child parametrisation, θ'
22: **end for** Loop for N generations

Bellman’s principle of optimality states; “An optimal policy has the property that whatever the initial state and initial decision are, the remaining decision must constitute an optimal policy with regard to the state resulting from the first decision.” [8] For the whole to be optimal, the constituent parts must also be optimal. If we can locate and combine sets of regional optima, we might therefore assume that the overall solution represents a combined global optimum.

Algorithm 5 $\text{compositeSelect}(\{A_i\}, A)$

Require: R

Number of STCA parameter regions

1: $\theta := \text{select}(A)$	Select a parametrisation, θ , from ‘overall’ archive, A (uniform or uniselect selection, section 3.1.2, page 48)
2: $\theta' := \text{nonRegionalParameters}(\theta)$	Extract the ‘non-regional’ parameters from parametrisation θ and assign them to child parametrisation θ'
3: for $i := 1$ to R do	
4: $\phi := \text{select}(A_i)$	Select a regional parameter subset, ϕ , from regional archive, A_i (uniform or uniselect selection, section 3.1.2, page 48)
5: $\theta' := \text{append}(\theta', \phi)$	Append the selected regional parameter subset, ϕ , to composite parametrisation θ'
6: end for	End splicing of child parametrisation, θ'
7: return θ'	Return composite parametrisation, θ'

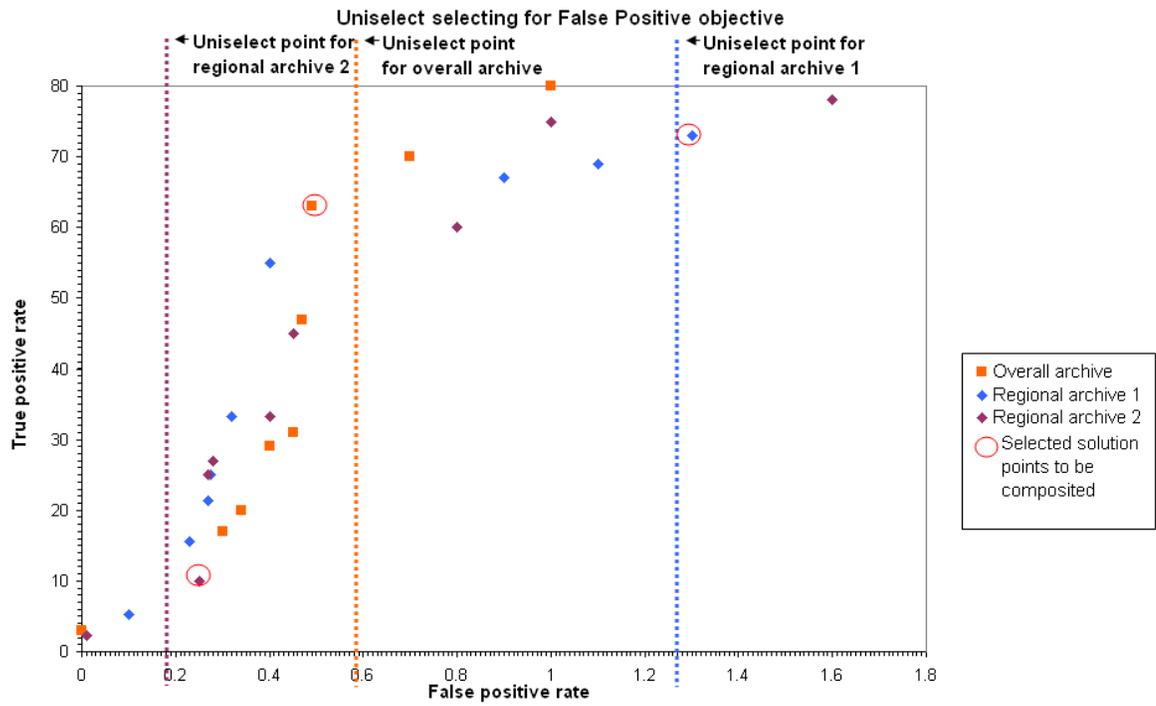
4.1.1 ‘Global’ Uniselect selection

With the introduction of multiple archives, we modify our selection scheme. Using the current options of uniform selection and Uniselect, parametrisations of varying regional TP and FP rates are composited together. ‘Global’ Uniselect seeks to combine parametrisations of similar regional TP and FP rates from each archive so that a better overall result may be achieved.

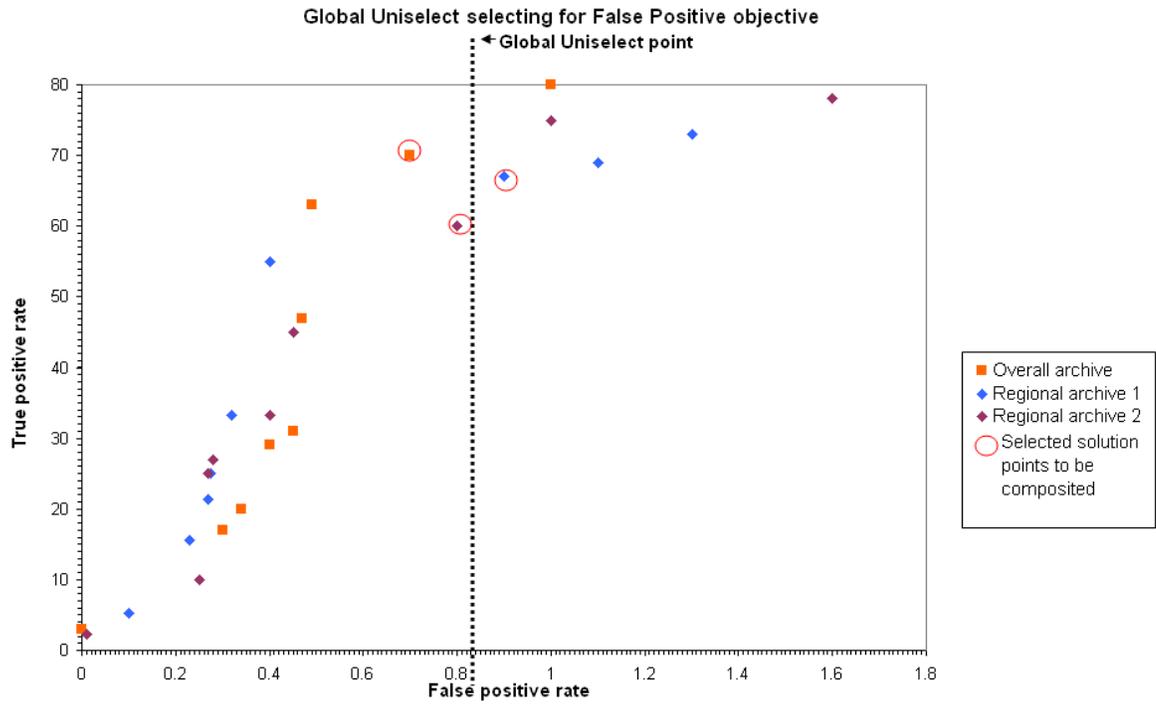
For instance, figure 4.3(a) shows selection from multiple archive fronts using just Uniselect. Each front has a different Uniselect selection point from which the nearest parametrisation is picked. Figure 4.3(b) shows Global Uniselect, again each front has a different bias towards certain parametrisations, however each selection is calculated from the same global position, resulting in a greater probability that ‘similar’ parametrisations will be picked for composition.

This selection scheme is designed to produce better quality child parametrisations upon composition that are less likely to be dominated by those already found in the ‘overall’ archive. That is, this scheme is less likely to mix low TP rate parametrisations from one regional archive with high TP rate parametrisations from another, which would bring the overall TP rate down when combined (and similarly for the FP objective).

We note that using Global Uniselect with the Single-Archive optimiser (algorithm 1) is equivalent to running it just using Uniselect.



(a) Uniselect selection on multiple archives.



(b) 'Global' Uniselect selection.

Figure 4.3: Schematic illustration of the operation of 'Global' Uniselect. See text for details.

Like the `compositeSelect` method (algorithm 5), `compositeUniselect` described in algorithm 6 is used to combine parameter subsets from all archives to create a new composite parametrisation. Algorithm 6, however, ensures that *global uniselect* is used to select parameter subsets with similar TP and FP rates across all archives.

Line 1 determines the probability, p , of end point selection and line 2 determines on which objective, j , selection should be based for the given iteration, n . At line 3 if the probability of end point selection is equal to or less than $pMin$ then, at lines 10 and 13 u will equal the minimum value belonging to the current objective’s range. If p is greater than $pMin$ but equal to or less than $pMin + pMax$ (line 5) then, at lines 10 and 13 u will equal the maximum value belonging to the current objective’s range. Otherwise, at lines 10 and 13 u will be uniformly drawn from a value *within* the current objective’s range.

At line 10 the value of u is calculated for the current objective, j , and ‘overall’ archive A . At line 11 the parametrisation, θ , with objective j closest to u is selected from archive A , the non-regional parameters from θ are then assigned to θ' . Lines 12 to 16 concern the selection and splicing of regional parameter subsets from $\{A_i\}$. At line 13 the value of u is calculated for the current objective, j , and current regional archive A_i . At line 14 the parameter subset, ϕ , with objective j closest to u is selected from archive A_i and assigned to ϕ' . ϕ' is then appended to composite parametrisation θ' at line 15. After all regional subsets have been appended (line 16), the composite parametrisation, θ' , is returned (line 17).

In summary, this multi-archive algorithm divides optimisation into nominally independent subsets and evolves each one on its own. In some respects this is similar to the concept of co-evolutionary algorithms [67, 65, 19, 46]. However, the multi-archive optimiser is not properly co-evolutionary because the subsets do not compete against one another in a predator-prey relationship, nor assuming independence, do they directly co-operate in any way. Each subset contributes to the overall archive, that is, parameter subsets contribute directly to an overall parametrisation and thus are evaluated in parallel for TP/FP rate. The subsets do not represent sub-species with independent, possibly distributed, fitness evaluation as in co-evolutionary frameworks. Individuals from other subsets do not remain fixed during evaluation, as credit assignment is not an issue.

Algorithm 6 Global Uniselect `compositeUniselect`($\{A_i\}, A$)

Require: R Number of STCA parameter regions
Require: n The current iteration number
Require: D Number of objectives (dimensions) e.g. TP and FP
Require: $pMin$ Probability of selecting min end point
Require: $pMax$ Probability of selecting max end point

1: $p := \text{rand}()$ The probability of selecting an end point
2: $j := n \bmod D$ Objective on which to perform Uniselect
3: **if** $p \leq pMin$ **then**
4: $v := 0$ Results in selection of the min end point
5: **else if** $p \leq (pMin + pMax)$ **then** Results in selection of the max end point
6: $v := 1$
7: **else**
8: $v := \text{rand}()$ Results in u being drawn uniformly
9: **end if**
10: $u := v \times (\max_{\theta \in A} f_j(\theta) - \min_{\theta \in A} f_j(\theta)) + \min_{\theta \in A} f_j(\theta)$ Scale u to the current objective's range, given archive A
11: $\theta' := \text{nonRegionalParameters}(\arg \min_{\theta \in A} |f_j(\theta) - u|)$ Select closest element, θ , to u as parent member and extract the 'non-regional' parameters from parametrisation θ , assigning them to child parametrisation θ'
12: **for** $i := 1$ to R **do**
13: $u := v \times (\max_{\phi \in A_i} f_j^i(\phi) - \min_{\phi \in A_i} f_j^i(\phi)) + \min_{\phi \in A_i} f_j^i(\phi)$ Scale u to the current objective's range, given regional archive A_i
14: $\phi' := \arg \min_{\phi \in A_i} |f_j^i(\phi) - u|$ Select closest element, ϕ , to u as parent member and assign it to ϕ'
15: $\theta' := \text{append}(\theta', \phi')$ Append the selected regional parameter subset, ϕ' , to composite parametrisation θ'
16: **end for** End splicing of child parametrisation, θ'
17: **return** θ' Return composite parametrisation, θ'

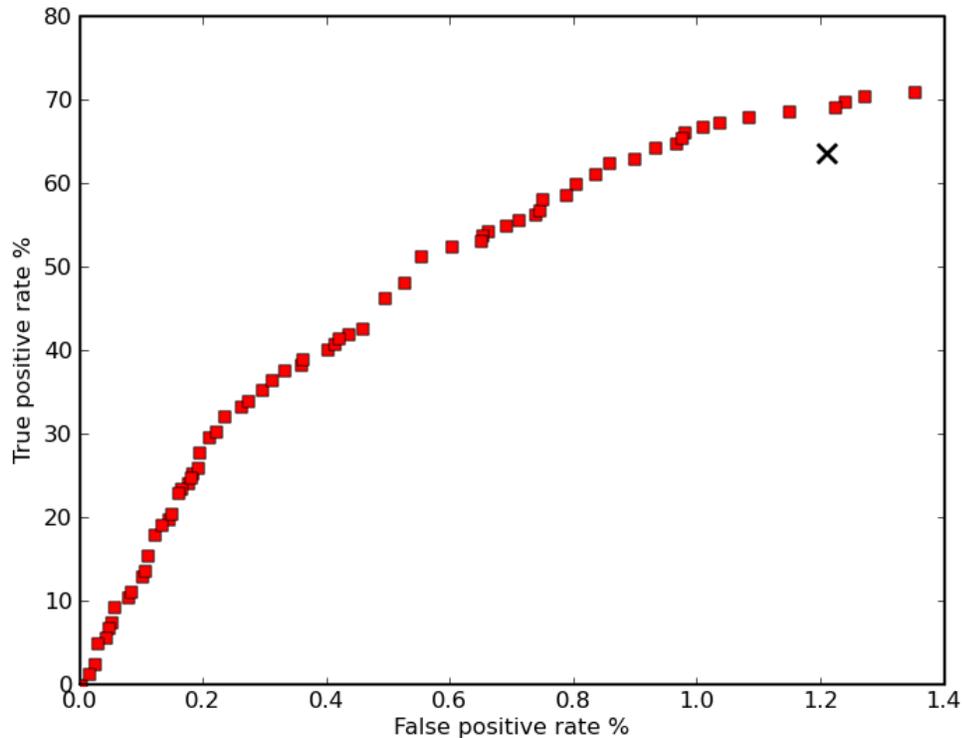


Figure 4.4: Multi-archive optimisation results, MACC dataset (6468 iterations). The black cross marks the NATS COP.

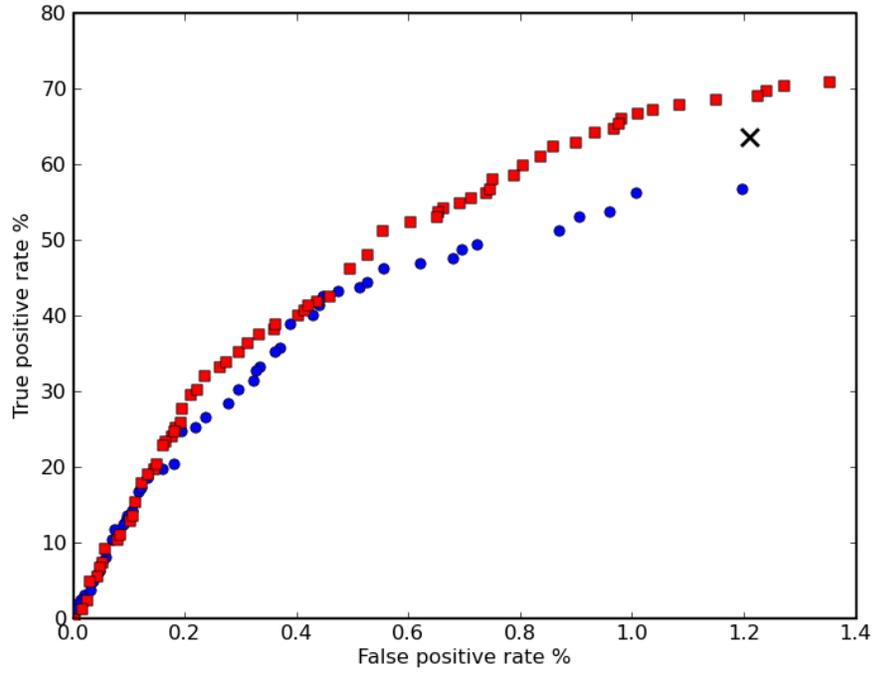
4.1.2 Results of the multi-archive optimiser

For comparison purposes this trial was run on the same dataset as the single-archive optimisation (see section 3.1.4); $\approx 70,000$ aircraft pairs, MACC July 2007.

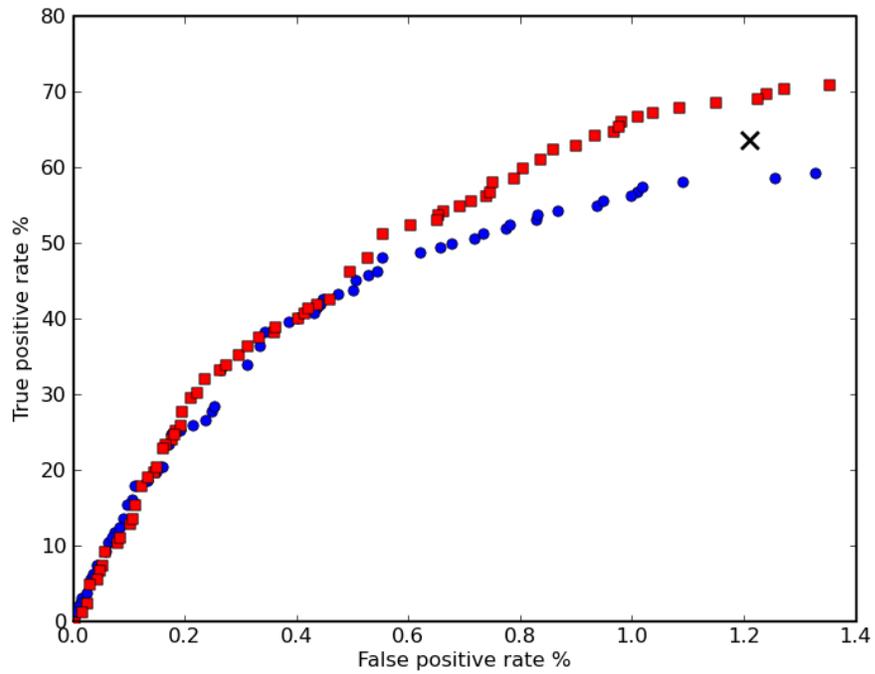
Figure 4.4 shows the archive contents after 6468 iterations. The optimiser has located an ROC consisting of 68 points ranging from 0.00% to 70.99% True Positive and 0.00% to 1.35% False Positive. The NATS manually tuned Current Operating Point (COP) was not included in the optimiser’s initial set, however it has been marked on the chart for comparison (highlighted as a black cross in figure 4.4, 63.58% TP and 1.21% FP). The initial set consisted of 100 parametrisations, each initialised with random parameter values.

The optimiser has located points that entirely dominate (are wholly better than) the COP. That is, for approximately the same tolerated FP rate the optimiser has located a parametrisation $\approx 5.6\%$ better in TP rate than the COP, for approximately the same tolerated TP rate the optimiser has located a parametrisation $\approx 0.3\%$ better in FP rate than the COP. Although the improvement over the COP is relatively small in percentage terms, the quantity of aircraft track pairs processed by the STCA system means that a significant reduction in the number of false alerts could be achieved while maintaining the current genuine alert rate.

In addition, perhaps of greater significance, the multi-archive optimiser has located a more optimal ROC in fewer than half the number of iterations that were required by single-archive mode. Figure 4.5(a) shows an overlay of the ROCs generated by single and multi-archive modes after 6468 iterations, and figure 4.5(b) shows that even after a further 7254 iterations



(a) Comparison of Single and Multi-archive modes after 6468 iterations each. Blue circles represent the single archive mode, red squares multi-archive, the cross indicates the manually tuned operating point.



(b) Comparison of Single and Multi-archive modes after single archive mode has been run for a further 7254 iterations (13722 iterations in total).

Figure 4.5: Comparison of Single and Multi-archive optimisation results, MACC dataset.

the single archive mode has still not caught up with the higher TP and FP rates located by the multi-archive optimiser.

4.1.3 Summary

We have seen how maintaining a set of elite archives, one for each regional parameter subset, can significantly reduce the number of evaluations required to generate a Pareto-optimal ROC for the NATS STCA system, thus increasing the speed of optimisation. The technique reduces the size of the search space and by maintaining multiple archives can effectively run sub-optimisations in parallel. In addition, the multi-archive approach can generate an ROC which entirely dominates that located by the single-archive optimiser (although at lower TP and FP rates the fronts virtually coincide). The trial has also demonstrated that the manually tuned operating point *can* be bettered starting from a random initialisation.

However, the multi-archive approach assumes regional parameter independence. In reality this is not the case due to cross-regional aircraft encounters and dataset region labelling discrepancies. This means that some noise is likely to be introduced to the candidate parametrisations, reducing the effectiveness of the optimisation approach. See section 4.3 on page 76 for a discussion of the causes of dependency and their effect.

4.2 The estimated-archive

In this section we expand on the multi-archive concept in the hope of further reducing the number of iterations required for optimal ROC generation. A method of recombining evaluated parameter subsets, to produce new ‘estimated’ parametrisations, is presented. The results are compared to previous single and multi-archive mode trials.

The ‘estimated’ archive expands on the multi-archive concept. The multi-archive mode is possible because we can obtain the number of True Positive and False Positive alerts found in each region of airspace, not just the overall STCA alerts rate, permitting optimisation on regional subsets of the STCA parameters.

However, the parameter subsets present in the regional archives can be recombined in many different ways, resulting in entirely new parametrisations. Estimated TP and FP counts of these new parametrisations can be calculated by adding together the associated regional TP and FP counts from the evaluated subsets in the regional archives. The estimated-archive would ideally represent the estimated ‘fitness’ of non-dominated parametrisations resulting from all possible recombinations of regional parameter subsets.

Of course generating every possible combination of parameter subsets is non-trivial for any reasonably large set of parametrisations. We can achieve a good approximation by simply substituting each regional subset of a newly evaluated parametrisation with the corresponding subset from each parametrisation already existing in the elite archive (performing a partial recombination). The new parametrisation may also be added to the estimated archive if it is not dominated by the existing members.

If the subsets of regional STCA parameters were truly independent, then the estimated TP and FP rates of each recombined parametrisation would be the same as if the parametrisations were evaluated for real on the STCA system. The estimated archive provides a means of generating hundreds of, potentially better, parametrisations without the need for evaluating each one. It should increase the speed of optimisation runs by reducing the number of evaluations necessary. In some respects the estimated-archive resembles a surrogate model (see section 2.4.5, page 41), permitting off-line evaluation of parametrisations. However, it is perhaps better thought of as a form of cross-over operator, identifying the ‘best’ regional combinations.

Figure 4.6 illustrates the partial recombination procedure. The top line depicts a newly evaluated parametrisation along side an existing member of the estimated archive. Below are possible combinations of inserting regional parameter subsets from the newly evaluated parametrisation into the estimated parametrisation. The estimated TP and FP rates are detailed alongside each of the new, recombined, parametrisations and are calculated by summing the associated regional TP and FP counts. In reality however, there are more than four STCA regions to recombine.

Newly evaluated solution:

		Regional parameter subsets				
		12	15	9	4	
LFCORI	8	8	11	7		<i>Evaluated overall</i> True Positive rate: 71/97 = 73% False Positive rate: 77/292 = 26%
LFFIRM	60	55	59	54		
LFSLOWL	15/30	19/20	7/12	30/35		
Regional TP:		4/99	10/80	23/40	40/73	
FP:		5/80	4/40	29/73		

Existing member of estimated archive:

		Regional parameter subsets				
		15	13	10	12	
LFCORI	10	5	8	10		<i>Estimated overall</i> True Positive rate: 64/97 = 66% False Positive rate: 46/292 = 16%
LFFIRM	60	65	59	60		
LFSLOWL	29/30	12/20	10/12	13/35		
Regional TP:		8/99	5/80	4/40	29/73	
FP:		8/99	5/80	4/40	29/73	

Recombination of new solution & existing estimated member, region 1:

		Regional parameter subsets				
		12	10	10	12	
LFCORI	8	5	8	10		<i>Estimated overall</i> True Positive rate: 50/97 = 52% False Positive rate: 42/292 = 14%
LFFIRM	60	65	59	60		
LFSLOWL	15/30	12/20	10/12	13/35		
Regional TP:		4/99	5/80	4/40	29/73	
FP:		4/99	5/80	4/40	29/73	

Recombination of new solution & existing estimated member, region 2:

		Regional parameter subsets				
		15	10	10	12	
LFCORI	10	8	8	10		<i>Estimated overall</i> True Positive rate: 71/97 = 54% False Positive rate: 51/292 = 17%
LFFIRM	60	55	59	60		
LFSLOWL	29/30	19/20	10/12	13/35		
Regional TP:		8/99	10/80	4/40	29/73	
FP:		8/99	10/80	4/40	29/73	

Recombination of new solution & existing estimated member, region 3:

		Regional parameter subsets				
		15	12	9	12	
LFCORI	10	5	11	10		<i>Estimated overall</i> True Positive rate: 61/97 = 63% False Positive rate: 65/292 = 22%
LFFIRM	60	65	59	60		
LFSLOWL	29/30	12/20	7/12	13/35		
Regional TP:		8/99	5/80	23/40	29/73	
FP:		8/99	5/80	23/40	29/73	

Recombination of new solution & existing estimated member, region 4:

		Regional parameter subsets				
		15	12	10	4	
LFCORI	10	5	8	7		<i>Estimated overall</i> True Positive rate: 81/97 = 84% False Positive rate: 57/292 = 20%
LFFIRM	60	65	59	54		
LFSLOWL	29/30	12/20	10/12	30/35		
Regional TP:		8/99	5/80	4/40	40/73	
FP:		8/99	5/80	4/40	40/73	

Figure 4.6: Creation of estimated parametrisations. See text for details.

Algorithm 7 Estimated-archive **Main()**

Require: N

Number of iterations

```
1:  $I := \text{initialise}()$             $I$  contains a set of parametrisations, some of which
                                may dominate others (section 3.1.1, page 47)
2:  $A := \emptyset$                  Initialise ‘overall’ archive as an empty set
3:  $E := \emptyset$                  Initialise ‘estimated’ archive as an empty set
4: for  $\theta$  in  $I$  do
5:    $\text{update}(A, \theta)$          Update overall archive,  $A$ , with  $\theta$  (algorithm 2, page 47)
6:    $\text{updateEstimates}(E, \theta)$  Update estimated archive,  $E$ , with  $\theta$  (algorithm 8)
7: end for                       End iteration through parametrisations in  $I$ 

8: for  $n := 1$  to  $N$  do
9:    $\theta := \text{select}(E)$        Select a parametrisation,  $\theta$ , from estimated archive,  $E$ 
                                (uniform or uniselect selection, section 3.1.2, page 48)
10:   $\theta' := \text{perturb}(\theta)$  Perturb parent to create a new child,  $\theta'$ ;  $\text{perturb}$  must ensure
                                that at least 1 parameter from each region is changed
11:   $(TP(\theta'), FP(\theta')) := STCA(\theta')$  Evaluate child’s TP and FP rates on STCA
12:   $\text{update}(A, \theta')$        Update overall archive,  $A$ , with child parametrisation  $\theta'$ 
                                (algorithm 2, page 47)
13:   $\text{updateEstimates}(E, \theta')$  Update estimated archive,  $E$ , with child
                                parametrisation  $\theta'$  (algorithm 8)
14: end for                       Loop for  $N$  generations
```

Algorithm 7 outlines the estimated-archive procedure more formally. At line 1 a set of parametrisations (I) is initialised and each member evaluated on the STCA system (see section 3.1.1 on page 47 for initialisation options). However, dominance of the members is not determined at this stage. Some of the parametrisations may therefore dominate others. Lines 2 and 3 initialise the ‘overall’ archive, A , and ‘estimated’ archive, E , as empty sets. Lines 4 to 7 update each of the archives with each parametrisation, θ , in set I . Note at line 6 a new method, ‘ updateEstimates ’, is used to update the estimated archive (see algorithm 8). The archives have now been initialised.

The evolutionary optimisation loop is entered at line 8. At line 9 a parent parametrisation, θ , is selected from the *estimated* archive, E , using either uniform or uniselect selection (see section 3.1.2, page 48). The parent parametrisation is then perturbed to create a new child parametrisation, θ' , at line 10 (see section 3.1.3 on page 50 for more on perturbation). Perturbation must ensure that at least one parameter from each regional subset is changed. At line 11 the child parametrisation, θ' , is evaluated on the STCA system. The TP and FP rates for the parametrisation are obtained. At line 12 the overall archive, A , is updated with θ' and at line 13 the estimated archive, E , is updated using the ‘ updateEstimates ’ method. The algorithm then loops back to line 8 and repeats the process N times.

Algorithm 8 `updateEstimates(E, θ')`

Require: R

Number of STCA parameter regions

```
1: for  $\theta$  in  $E$  do
2:   for  $i$  in  $R$  do
3:      $\phi := \text{nonRegionalParameters}(\text{getParent}(\theta))$       Extract the ‘non-regional’
                                                                parameters from a parent parametrisation
                                                                of  $\theta$  and assign them to estimated parametrisation  $\phi$ 
4:     for  $j$  in  $R$  do
5:       if  $i = j$  then
6:          $\chi := \text{extractRegion}(\theta', i)$       Extract parameter subset,  $\chi$ , corresponding to
                                                                region  $i$  from parametrisation  $\theta'$ 
7:       else
8:          $\chi := \text{extractRegion}(\theta, j)$       Extract parameter subset,  $\chi$ , corresponding to
                                                                region  $j$  from parametrisation  $\theta$ 
9:       end if
10:       $\phi := \text{append}(\phi, \chi)$       Append regional parameter subset,  $\chi$ , to composite
                                                                estimate parametrisation  $\phi$ 
11:    end for
12:     $\text{update}(E, \phi)$       Update estimated archive,  $E$ , with estimated
                                                                parametrisation  $\phi$  (algorithm 2, page 47)
13:  end for
14: end for      End iteration through parametrisations in  $E$ 
15:  $\text{update}(E, \theta')$       Update estimated archive,  $E$ , with evaluated parametrisation  $\theta'$ 
                                                                (algorithm 2, page 47)
```

The `updateEstimates` method described in algorithm 8 is used to generate multiple estimated parametrisations by combining each existing member of the estimated archive, E , with new evaluated parametrisation θ' . The estimated archive is updated with each new estimated parametrisation; this means that for every evaluated parametrisation, θ' , archive E will be updated with multiple estimates.

At line 1 the algorithm begins to iterate through each existing member, θ , of estimated archive E . Lines 2 and 4 form a nested loop designed to combine regional parameter subsets from θ and θ' . In order to maintain diversity in non-regional parameters `getParent` selects one of the parents of the parametrisation θ . Line 3 assigns parametrisation ϕ ‘non-regional’ parameters selected in a uniform manner (using `getParent`) from the set of parent parametrisations belonging to θ . Note that the non-regional parameters of θ itself may also be selected. At line 6 if region iteration i equals region iteration j then the parameter subset corresponding to region i from parametrisation θ' is assigned to χ . If region iteration i is not equal to region iteration j then the parameter subset corresponding to region j from parametrisation θ is assigned to χ (line 8). χ is then appended to parametrisation ϕ (line 10). Once estimated parametrisation ϕ has a complete set of regional parameters it is updated to the estimated archive, E (line 12). The process of combining regional parameter subsets then starts again at line 2 for the next i in R . When iteration through each existing member of E has completed (line 14), parametrisation θ' itself is updated to archive E (line 15).

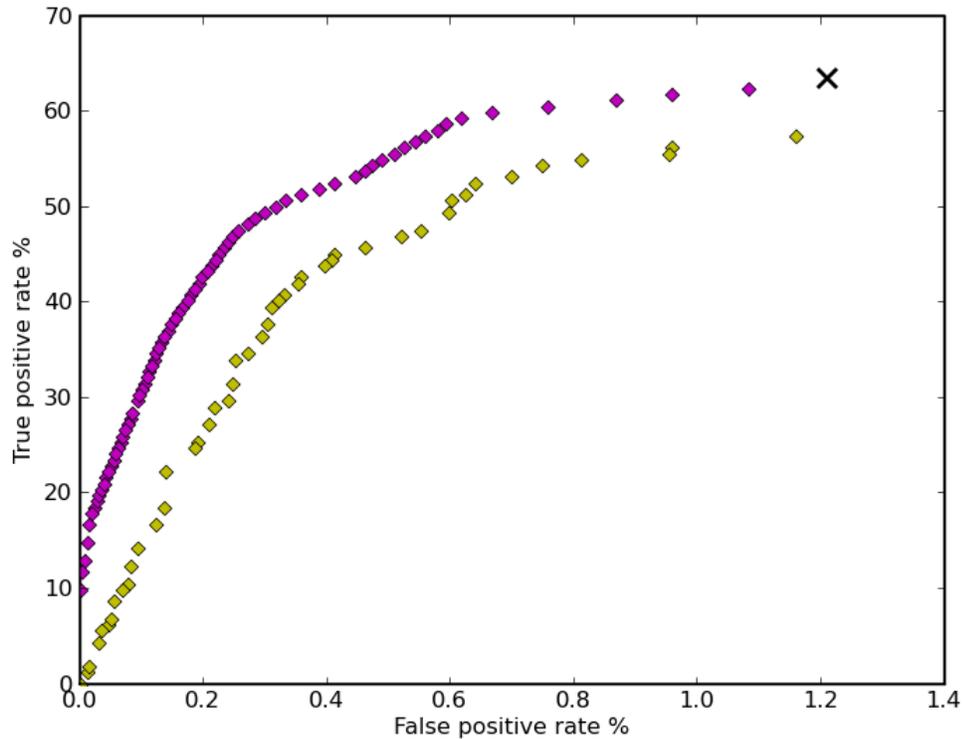


Figure 4.7: Estimated-archive optimisation results, MACC dataset (12677 iterations). Yellow diamonds represent the evaluated ROC, purple diamonds the ‘estimated’ ROC. The NATS COP has been marked as a black cross.

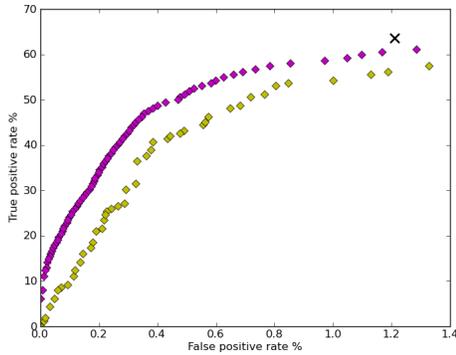
4.2.1 Results of the estimated-archive

Figure 4.7 shows the archives’ contents after 12677 iterations on two weeks ($\approx 70,000$ aircraft pairs) MACC July 2007 data. The optimiser has located an actual (evaluated) ROC consisting of 46 points ranging from 0.00% to 57.41% True Positive and 0.00% to 1.16% False Positive (yellow diamonds). An ‘estimated’ ROC consisting of 77 points, lies from 9.88% to 62.35% TP, 0.00% to 1.08% FP (purple diamonds). The trial was initialised with 100 randomised parametrisations. The NATS manually tuned Current Operating Point (COP) has been marked as a black cross (63.58% TP, 1.21% FP). In this case the NATS COP has not been dominated.

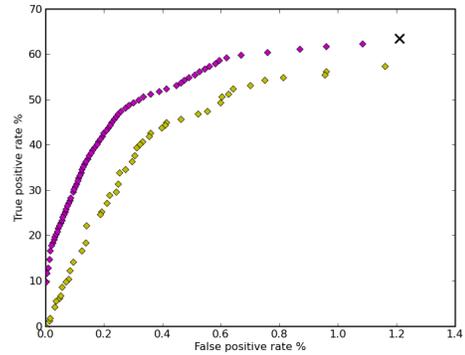
The archive of recombined solution estimates (purple) contains almost double the number of solutions in the evaluated (yellow) archive. This demonstrates the ability of the estimated archive to assess numerous parameter combinations off-line, potentially identifying more optimal configurations prior to actual evaluation. It has led to a greater density of solutions across the estimated front, as only slight variations in regional TP/FP rate generate non-dominating positions when re-combined to give an (estimated) overall TP/FP value.

Each element of the evaluated archive corresponds to a (perturbed) element of the estimated archive. Some of the correspondences between parent and child are clear from the figure, particularly at higher TP/FP rates. This gives us an estimate of the error between

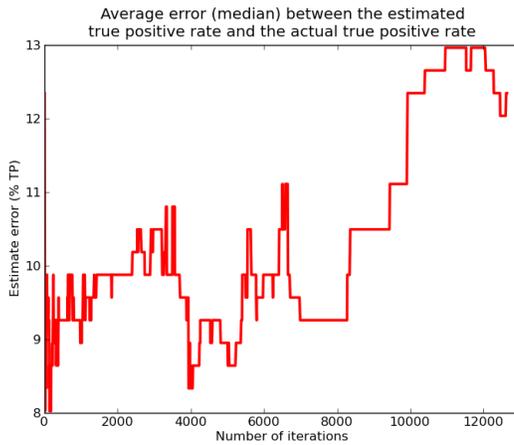
the estimated and evaluated archives of $\approx 12\%$ TP and $\approx 0.15\%$ FP after 12677 iterations. This error increases over the generations, referring to figures 4.8(a) and 4.8(b) we can see that the discrepancy between estimated and actual TP/FP rates has subtly risen. Figures 4.8(c) and 4.8(d) further highlight this trend. These show an estimate of the change in the error with generation. For TP, the approximate error is calculated by taking the median difference in TP between a point in the ‘true’, evaluated archive and the point in the estimated archive with the closest FP; vice versa for FP.



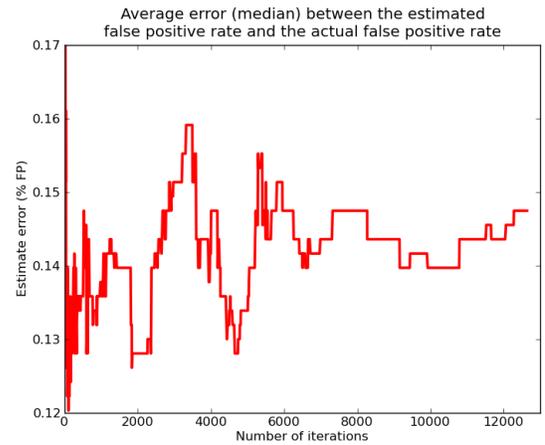
(a) Estimated-archive after 2000 iterations.



(b) Estimated-archive at trial completion (12677 iterations in total).

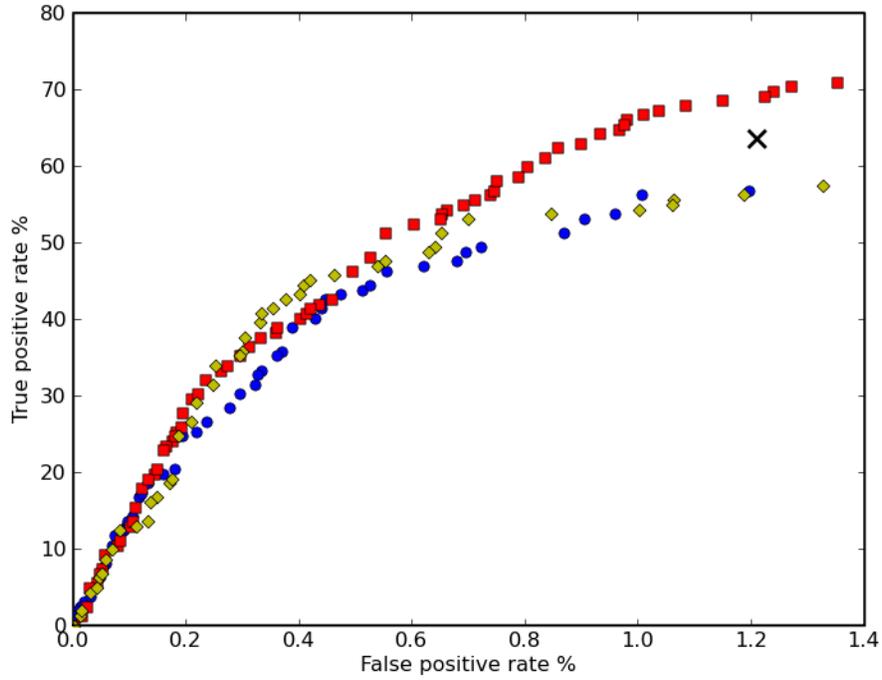


(c) Estimate of the average error (median) between the estimated true positive rate and the actual true positive rate.

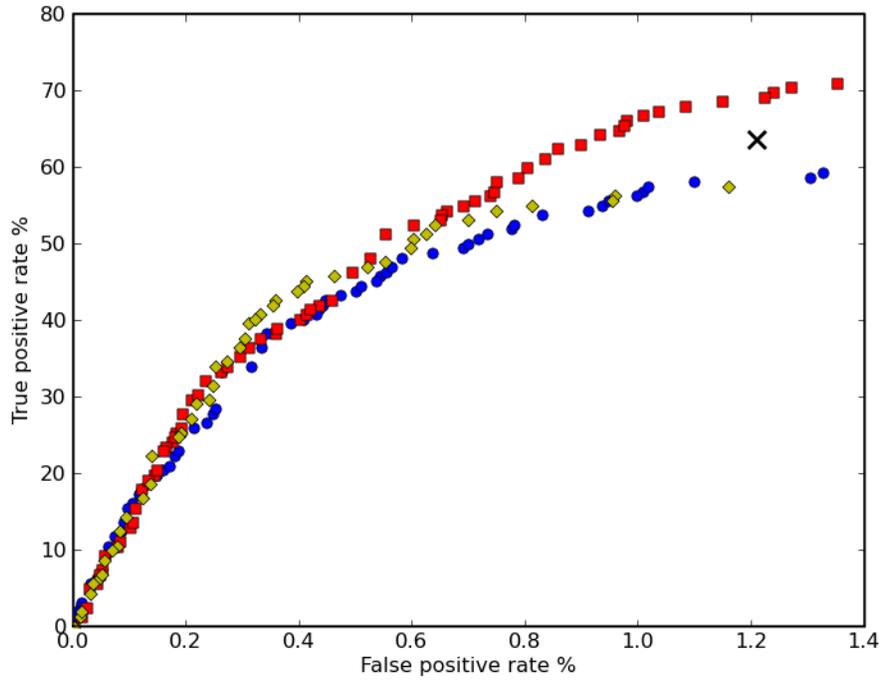


(d) Estimate of the average error (median) between the estimated false positive rate and the actual false positive rate.

Figure 4.8: Comparison of the divergence between evaluated and ‘estimated’ archives on the MACC dataset. In figures 4.8(a) and 4.8(b), yellow diamonds represent the evaluated ROC, purple diamonds the ‘estimated’ ROC. The NATS COP has been marked as a black cross. In figures 4.8(c) and 4.8(d), the red line represents an estimate of the average error (median) in TP and FP between points on the estimated ROC (purple) and those on the evaluated ROC (yellow).



(a) Comparison of Single, Multi and Estimated-archive modes after 6468 iterations each. Blue circles represent the single archive mode, red squares multi-archive, yellow diamonds estimated-archive, the cross indicates the manually tuned operating point.



(b) Comparison of Single, Multi and Estimated-archive modes after single and estimated-archive modes have been run for a further 6209 iterations (12677 iterations in total).

Figure 4.9: Comparison of Single, Multi and Estimated-archive optimisation results on the MACC dataset.

Even after 12677 iterations the estimated archive still lies substantially ahead of the actual ‘overall’ archive of *evaluated* parametrisations. If the estimated archive were working correctly it is expected that the ‘overall’ archive would eventually converge upon the estimated archive’s position as the optimal ROC is located. This however, has not occurred, while the lack of convergence may imply that the optimiser simply has not been run for long enough, further analysis suggests otherwise.

Comparing the evaluated front to those generated by previous trials (figure 4.9(a)), we see that it has largely dominated the single-archive mode and has managed to dominate the multi-archive mode in one place ($\approx 36\%$ to $\approx 45\%$ TP, $\approx 0.3\%$ to $\approx 0.45\%$ FP). Referring to figure 4.9(b) we see that the estimated mode’s evaluated front still dominates single archive, however single archive has begun to catch up. The dominance over multi-archive mode has not greatly improved.

The estimated mode’s evaluated front has not really progressed in the subsequent 6209 iterations; it seems to have stalled. This indicates that although in early iterations it has a speed advantage over single archive mode (converging faster), the estimated archive appears to eventually inhibit further progress. It should also be noted that, if the multi-archive mode were to be allowed to run for longer it is likely that it too would locate the area of the ROC currently dominated by the estimated-archive’s evaluated front.

It is suggested that the eventual inhibition of the estimated-archive is caused by a greater degree of interaction between regional parameters than was previously anticipated. Dependencies between regional parameters can introduce noise reducing the effectiveness of parametrisation perturbation. As the optimisation has progressed the estimated fitness’s of parametrisations selected from the estimated archive (the purple diamonds in figure 4.7) are so incongruent with the actual evaluated fitness’s that new parametrisations entering the evaluated archive (the yellow diamonds) are always dominated by existing members. Exploration of the search space is inhibited as the ‘gene-pool’ stagnates due to selection from ‘corrupted’ parent members in the estimated archive.

4.2.2 Summary

Although initial progression of the ROC is encouraging, further advancement of the front appears to stall after only a few thousand iterations. At low TP and FP rates the ROC is roughly equivalent to both single and multi-archive modes, however estimated-archive mode fails to dominate multi-archive’s front at higher TP and FP rates, even after it has been run for double the number of iterations. We speculate that noise, introduced by dependencies between regional parameters, inhibits the estimated-archive’s development. The multi-archive optimiser is less affected by this because parametrisations are evaluated for real *each* iteration and so there is not the same accumulation of misleading estimates. Therefore, the best method investigated so far, in terms of ROC optimality and speed, is still ‘multi-archive mode’. This remains as the ‘benchmark’ to be exceeded.

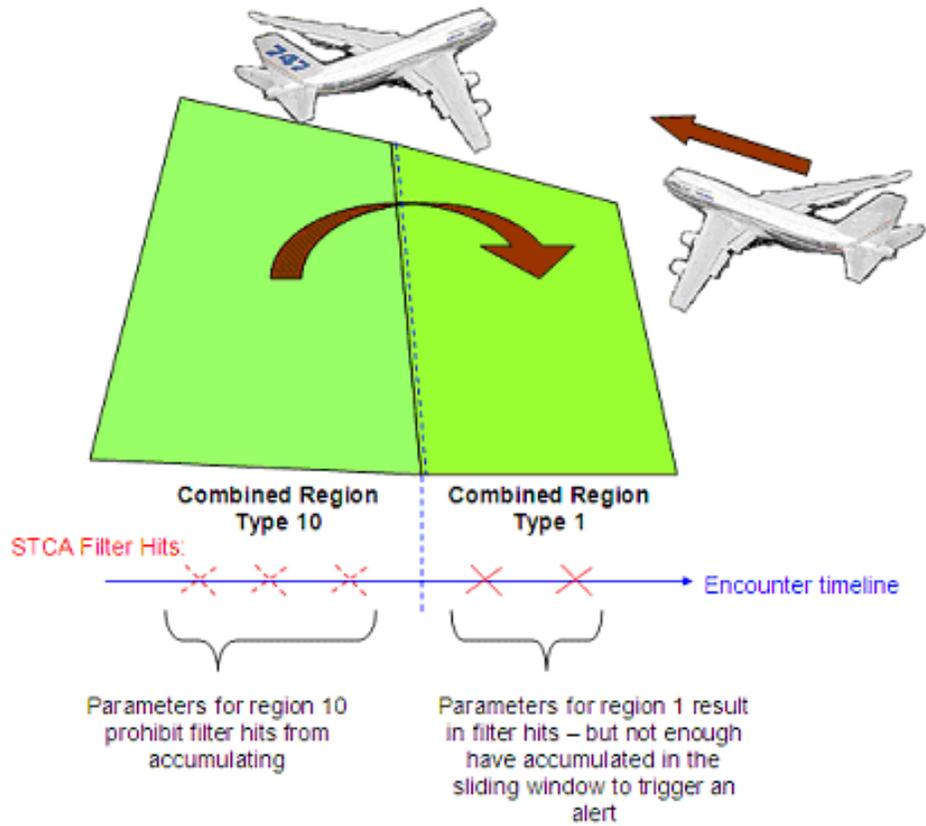


Figure 4.10: Illustration of how the STCA filter hits can cause regional dependencies in STCA parameter sets. The figure shows an aircraft crossing from one region to another and coming into conflict with a second aircraft.

4.3 Discussion of efficacy

The regional and estimated archives methods are only practical due to the near-independence of regional STCA parameters, allowing them to be divided into separate subset optimisations (see sections 4.1, 4.2). If all non-regional parameters were disabled for optimisation (i.e. were set to fixed values in all parametrisations), then one could presume that the regional parameters are completely independent of one another; however, this is not necessarily the case.

Any dependencies between regional parameters can introduce noise reducing the effectiveness of parametrisation perturbations and, in the case of the estimated-archive, eventually inhibiting further ROC progression. The next sections examine some of the causes of dependency between regional parameters and looks at ways of reducing their impact.

4.3.1 Cross-regional aircraft pairs & parameter independence

So that separate rules can be applied to different sections of airspace, STCA divides the airspace into regions. This allows parts of the system to have different behaviours, based on the nature of air traffic patterns for a specific zone (see figure 2.4 on page 18).

Fundamentally, STCA deals in pairs of aircraft; however, during the course of a conflict, one or more of the aircraft may cross region boundaries. The system can only select one rule set to work from at any one time, so a matrix is used to calculate the combined region type of the two aircraft (see figure 2.5 on page 18).

As aircraft cross boundaries the combined region type may change, thus an alerting scenario may start in one region but be resolved in another. Which region the scenario is assigned when alert statistics are output by STCA depends on at which point along the aircraft tracks the alert was triggered and this of course will vary depending on how the STCA parameters/rules for the region have been configured.

Although, strictly speaking, the regional parameters are independent, the ‘filter hits’ associated to each aircraft pair are maintained between regions. For example; consider aircraft travelling from one region to another during an encounter, causing the pairs combined region type to change mid-encounter (figure 4.10). The combined region type changes from ‘region 10’ to ‘region 1’. If STCA has a ‘sliding-window’ setup to confirm an alert after 4 cycle hits in region 1, but the parameters in region 10 have been configured in such a way that the filter hits do not occur, then as the combined region type switches to region 1 there are insufficient hits present to trigger an alert.

Thus, changing parameters controlling one region can affect the alert rate of encounters occurring in another. Fortunately, the proportion of cross-regional encounters in a dataset falling into this category is relatively small, minimising the impact on TP and FP rates for the regional archives.

4.3.2 Dataset regional labelling discrepancies

As described in section 2.3.3, aircraft pair datasets are semi-manually categorised into ‘wanted’ and ‘unwanted’ alerts and assigned a region type by NATS’ staff. A situation can arise where the dataset used during optimisation of STCA has aircraft pairs assigned one region type, and the statistics output from STCA indicate another. In addition, the region assigned to an alerting pair in STCA may vary depending on when the alert was triggered. This means that an accurate representation of regional true and false positives cannot be obtained without first aligning the results; by looking up pair numbers in the STCA output and obtaining the region as assigned in the original dataset, accurate alert totals can be obtained for each region.

However, while this corrects the discrepancy in totalling alerts, it does not correct which regional parameters were ultimately used for the combined region type. Put another way, by optimising parameters for, say, ‘region 2’, it might be assumed that the corresponding TP and FP rates for this region alone would be affected. If however, an encounter pair categorised in the dataset as occurring in ‘region 1’, is determined by STCA to be in region 2, then STCA will be using the region 2 parameters. Thus, because the pair was determined to be in region 1 in the original dataset, the TP and FP rates for region 1 will be affected by changes made to region 2’s parameters.

Unfortunately, this discrepancy can lead to quite a large variability in the regional archives’ Pareto fronts, reducing the benefits gained from using regional and estimated archives to speed

up the overall optimisation process. However, it should be noted that the quality of individual parametrisations in the ‘overall’ archive is not affected, only the speed at which the optimal ROC is generated.

4.3.3 Correcting categoriser region-labelling discrepancies

During dataset compilation a categorisation tool (‘Category_V3’) is used to process historical STCA encounters and assign ‘Cat’ 2s and 4s to each aircraft pair (see section 2.3.3 on page 19). This is done as a precursor to further manual review for ‘Cat’ 1s and 3s. The tool is used to speed up the time-consuming task of dataset compilation.

Cat 1 - Risk of collision (alert wanted)

Cat 2 - Significant loss of separation (alert desirable)

Cat 3 - Level off with risk (alert understandable but unwanted)

Cat 4 - Nuisance (alert undesirable)

Cat 5 - Bad data (invalid for optimisation)

Previously only the category assignments had been of significance to optimisation, however, with the introduction of the Multi-archive optimiser the regional labels given to each aircraft pair by Category_V3 have become more relevant.

Manually amending Category_V3 assignments to agree with STCA

In a test scenario on two weeks of MACC data (July 2007, $\approx 70,000$ aircraft pairs) we compiled a regional discrepancies matrix for 57 different STCA parameter configurations (derived from a Pareto front of parametrisations from an earlier optimisation run on the same dataset). By implementing a diverse set of STCA configurations we hoped to obtain a better picture of which aircraft pairs STCA commonly mis-assigns regions to (in respect of the *dataset* (Category_V3) region designation).

The results were as follows:

Total number of aircraft pair alerts:	739
Number of pairs for which the STCA categorisation is <i>always</i> different from the Category_V3 region designation:	377
Number of pairs for which the STCA and Category_V3 region designation <i>always</i> concur:	334
Number of pairs for which the STCA and Category_V3 region designation <i>sometimes</i> (but not always) concur: (Most likely due to cross-regional aircraft encounters)	28

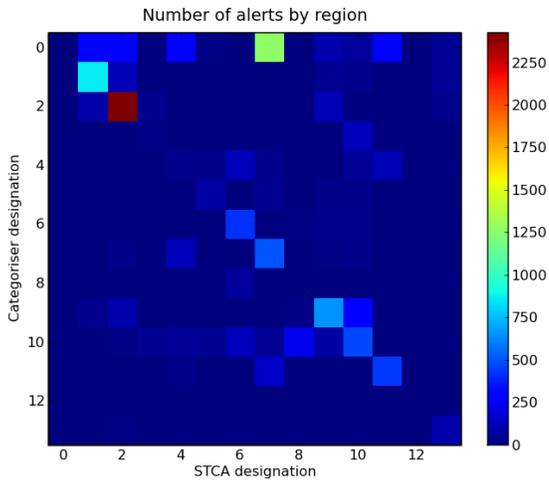
Figure 4.11: Category_V3 discrepancies. See also figure 4.12.

Note that results only show *alerting* aircraft pairs; STCA does not output information on pairs that do not alert.

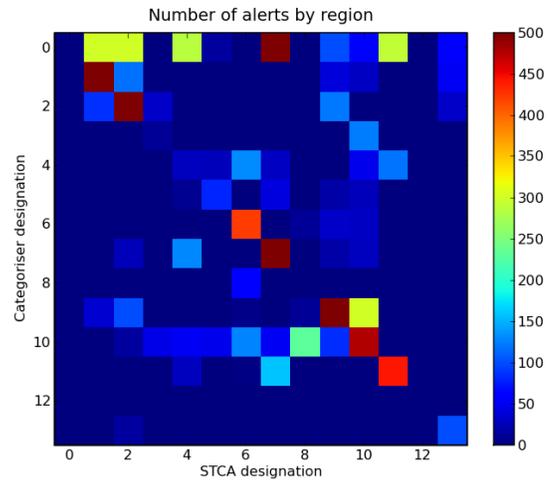
Of the 337 pairs that always alerted in different regions from the dataset only 61 alerted in *multiple* regions. Therefore, 276 pairs were simply ‘out-right’ mis-assigned; that is, throughout all STCA parameter configurations the alerts always occurred in the same STCA assigned region, which was different the dataset’s assigned region.

By manually re-labelling the 276 cases of ‘mis-assignment’, the number of ‘correctly’ allocated regions can be increased to 610 out of 739 aircraft pairs. It is expected that this will reduce the amount of noise introduced to the multi-archive perturbation of parameters and increase the quality of regional Pareto fronts, improving optimisation.

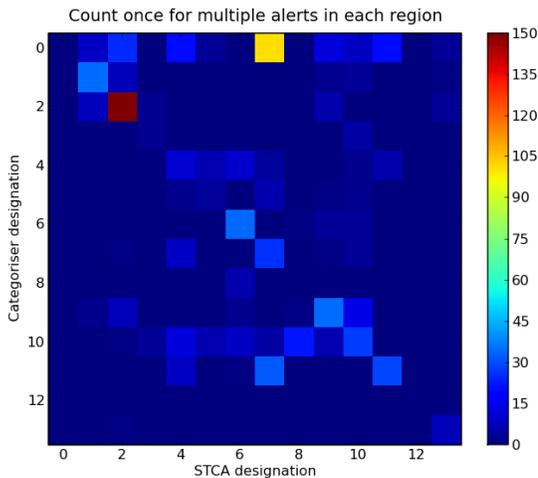
Clearly, the STCA assigned regions are not definitive and are still subject to cross-regional discrepancies however, it is interesting to note the high number of outright mismatches between STCA and Category_V3. This is highlighted in figure 4.12 by the high degree of scattering in “confusion matrix” plots for Category_V3 and STCA region assignment. The colour of each square represents the number of track pairs categorised by Category_V3 and STCA to the regions indicated by the row and column coordinates. If the region designations between STCA and Category_V3 were identical we should see a perfectly diagonal line plotted from top left to bottom right, however this is not the case.



(a) STCA region assignments for alerting pairs vs. Category_V3 region assignments in the dataset.



(b) STCA region assignments for alerting pairs vs. Category_V3 region assignments in the dataset with count restricted to a maximum of 500 to highlight smaller figures.

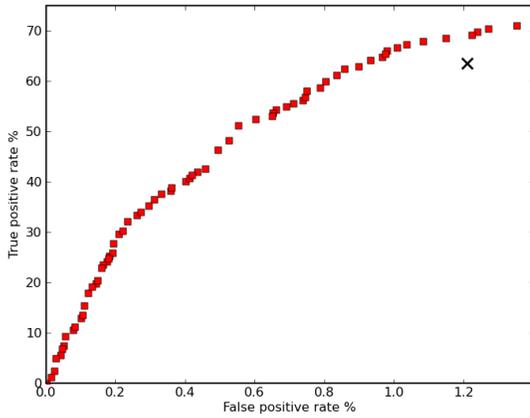


(c) STCA region assignments for alerting pairs vs. Category_V3 region assignments in the dataset with counts only incremented for each unique result (region combination per aircraft pair) across all 57 parameter configurations.

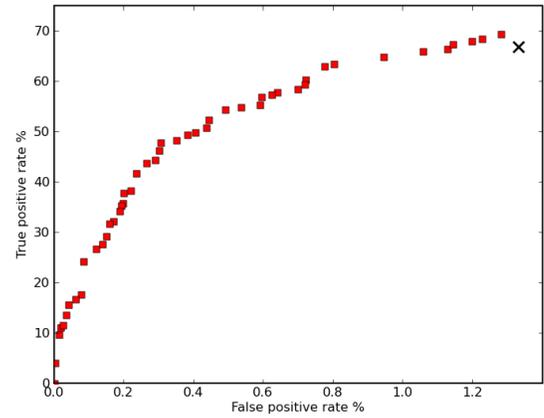
Figure 4.12: Category_V3 discrepancy matrices. Each matrix indicates the number of region assignments for alerting pairs for a particular categoriser designation and the region in which they actually alerted.

Results of manually correcting the region-labelling discrepancies

Re-running the Multi-archive and Estimated-archive trials with the new *manually* corrected regions dataset produced the following results. For comparison purposes results are shown at 6468 iterations.

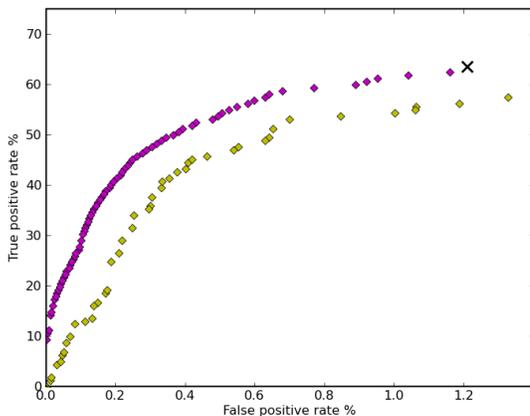


(a) Multi-archive 6468 iterations, original dataset.

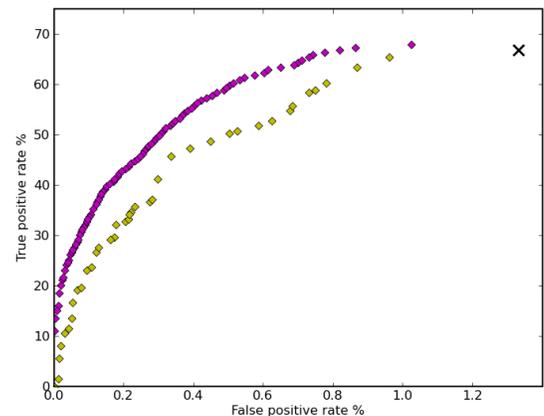


(b) Multi-archive 6468 iterations, corrected dataset. Note the change in COP after correction.

Figure 4.13: Multi-archive optimisation results, MACC dataset with corrected region assignments.



(a) Estimated-archive 6468 iterations, original dataset. (Purple diamonds are the 'estimated' front, yellow diamonds the *evaluated* 'overall' front).

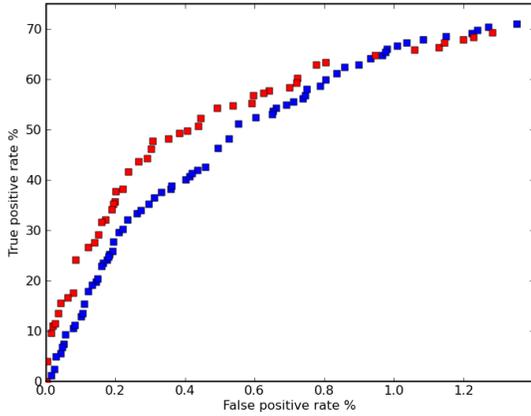


(b) Estimated-archive 6468 iterations, corrected dataset. Note the change in COP after correction.

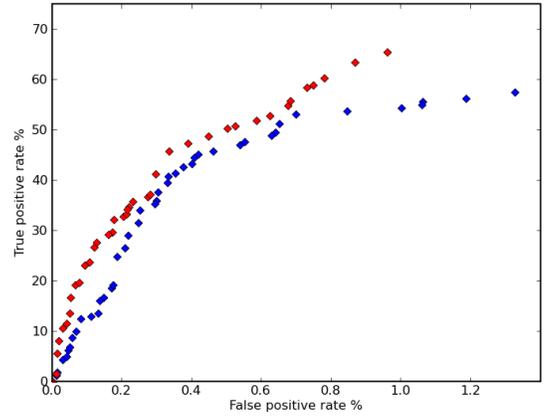
Figure 4.14: Estimated-archive optimisation results, MACC dataset with corrected region assignments.

As we can see from figure 4.13, correcting the region labelling discrepancies has an effect on optimisation speed. The regional archives work more efficiently (the regional TP and FP rates associated to regional parameters are more accurate), producing an ROC which almost totally dominates the uncorrected run (figure 4.13(a)) in an equivalent number of iterations (figure 4.13(b)). See also figure 4.15(a) for overlay of results.

Similarly, less noise has been introduced to the estimated archive (figure 4.14); the ROC



(a) Multi-archive 6468 iterations, original (blue squares) vs. corrected (red squares) dataset.



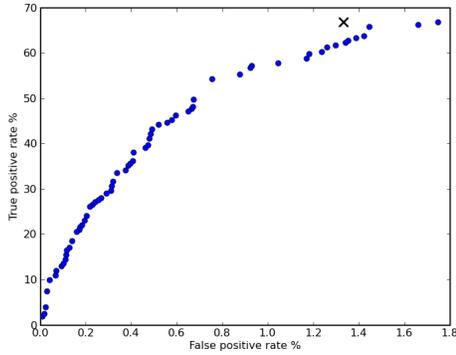
(b) Estimated-archive 6468 iterations, *evaluated* front, original (blue diamonds) vs. corrected (red diamonds) dataset.

Figure 4.15: Comparison of convergence between the original and corrected region assignments for the MACC dataset, using the Multi-archive (4.15(a)) and Estimated-archive (4.15(b)) optimisers.

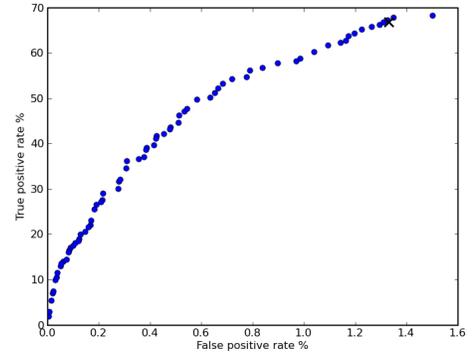
has managed to progress further before inhibition (figure 4.14(b)), totally dominating the uncorrected trial (figure 4.14(a)). See also figure 4.15(b) for overlay of results. However, even with dataset correction, the estimated archive performs poorly compared to multi-archive mode. Inhibition still occurs due to the dependencies associated with non-regional parameters and the remaining uncorrected labelling discrepancies caused by cross-regional encounters.

For the purposes of completeness the single-archive optimiser (blue circles) was also re-run on the corrected dataset. At 6468 iterations (figure 4.16(a)) it has not been able to locate the NATS operating point. However, it is interesting to note that if left to run for a further 7254 iterations (to 13722, figure 4.16(b)), for the first time, it *has* been able to locate the operating point. This further highlights the importance of correct region labelling in encounter pairs. Even when the regional archives are not in use, the reduction of noise in the dataset leads to more efficient optimisation. Figure 4.16(c) highlights the difference in ROC resulting from the corrected dataset after 13722 iterations.

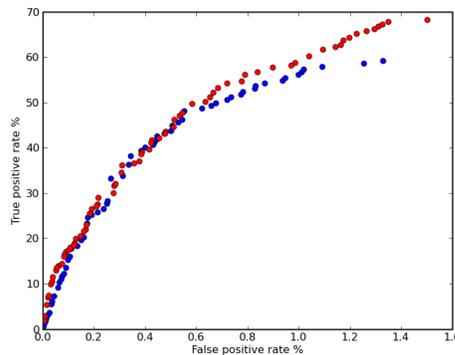
A change in the NATS current operating point may be observed between original (figures 4.13(a) and 4.14(a)) and corrected (figures 4.13(b) and 4.14(b)) datasets. This occurs due to track pairs previously categorised as ‘region 0’ being re-labelled (see section 4.3.4, page 87), resulting in inclusion of additional pairs not previously considered in TP and FP totals of the original datasets.



(a) Single-archive 6468 iterations, corrected dataset.



(b) Single-archive 13722 iterations, corrected dataset. Note the COP has been located.



(c) Single-archive 13722 iterations, original (blue circles) vs corrected (red circles) datasets.

Figure 4.16: Comparison of convergence between the original and corrected region assignments, using the Single-archive optimiser on the MACC dataset.

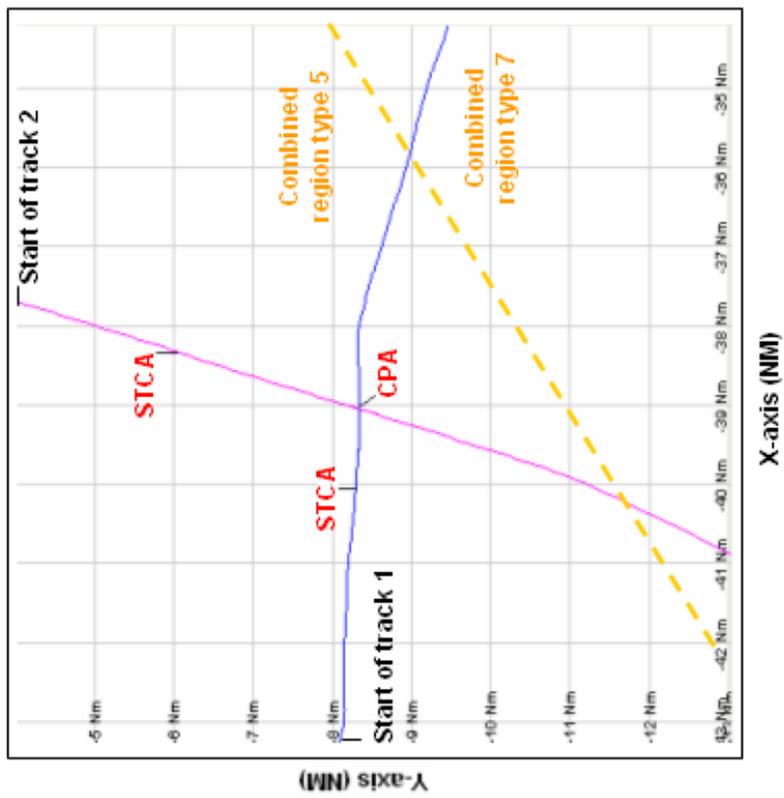
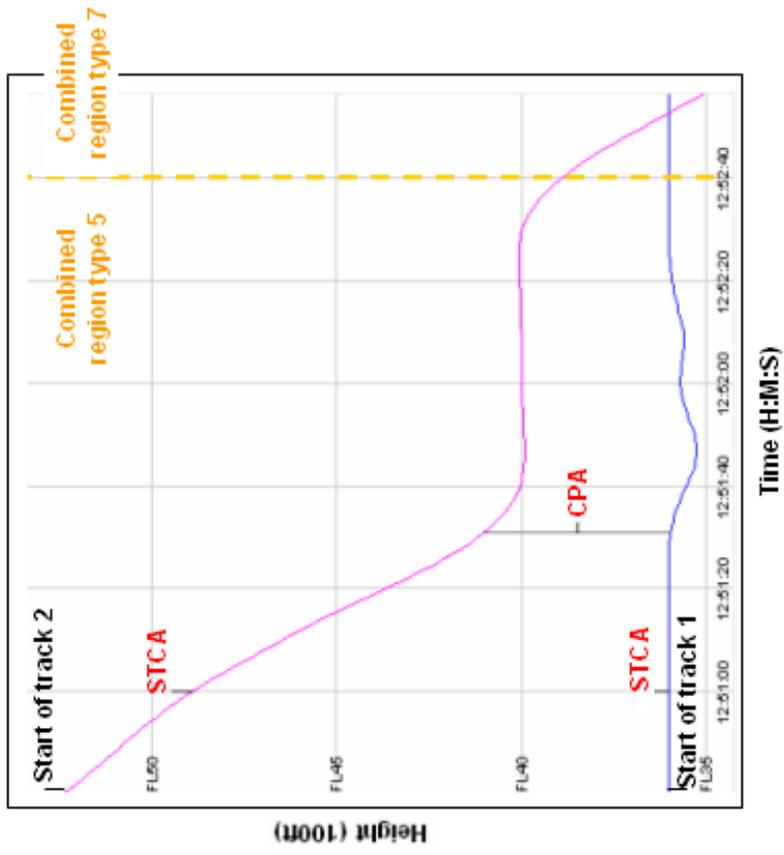
Amending Category_V3 to define region type based on CPA

Manual re-labelling of assigned regions in the dataset is a time consuming task and not really practical for routine STCA optimisations. We therefore investigated why the automated categoriser (Category_V3) assigns regions so differently to STCA.

It was found that Category_V3 reports the last computed combined region type for a track pair, i.e. the region as defined at the end ¹ of the aircraft tracks is assigned to the pair. While it is accepted that, due to the nature of the problem, Category_V3 can never predict exactly which region STCA will assign (as that depends on the STCA parameters), it was suggested that using the Closest Point of Approach (CPA) would produce significantly fewer outright mismatches than simple assignment based on the end of tracks, because the combined region type at the CPA is most likely to be the point near which STCA alerts for a track pair.

For example, figure 4.17 shows a fictional encounter pair scenario which crosses an STCA region boundary. The left graph shows the encounter in the horizontal plane (X-Y chart), while the right plots aircraft altitudes over time. The CPA has been marked in red and

¹The last point that the aircraft positions were determined to pass the coarse filtering test.



- Aircraft track 1
- Aircraft track 2
- STCA Airspace region boundary

Figure 4.17: Example aircraft encounter pair, categorisation based on CPA vs. end of track.

the likely point at which STCA would alert labelled on each track. The aircrafts' combined region type at the time of alert is 'region 5'. If we were to categorise the track pairs using Category_V3 (which uses the end of the tracks to assign region) then the combined region type would be determined as 'region 7', causing discrepancy. However, if we were to assign dataset regions based on CPA then 'region 5' would have correctly been labelled.

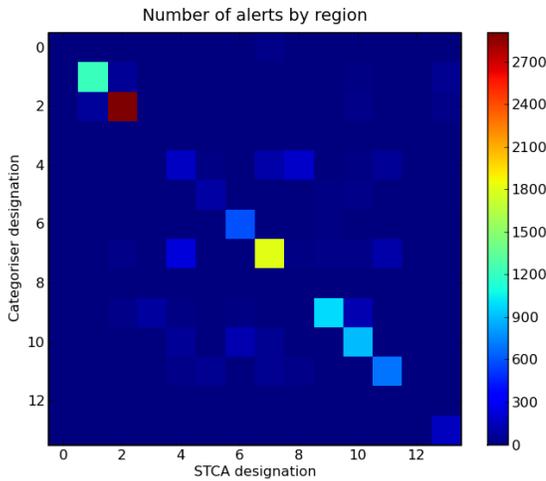
Subsequently, a new version of the categoriser ('Category_V3B') was created that uses CPA instead of 'end of track pair' to assign regions. As the following results show, use of CPA significantly reduced the number of discrepancies between STCA and the categoriser.

Total number of aircraft pair alerts:	739
Number of pairs for which the STCA categorisation is <i>always</i> different from the Category_V3 region designation:	165
Number of pairs for which the STCA and Category_V3 region designation <i>always</i> concur:	525
Number of pairs for which the STCA and Category_V3 region designation <i>sometimes</i> (but not always) concur: (Most likely due to cross-regional aircraft encounters)	49

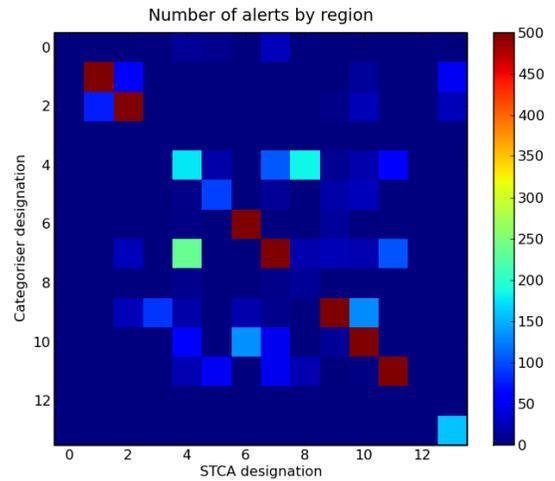
Figure 4.18: Category_V3B discrepancies. See also figure 4.19.

Of the 165 pairs that always alerted in different regions from the dataset 61 alerted in *multiple* regions. This means that the number of outright mismatches was decreased by 212 (from 316 for Category_V3 to 104 for Category_V3B). 'Correctly' categorised regions increased by 191 (from 334 for Category_V3 to 525 for Category_V3B). Understandably the number of pairs that always alert in both the dataset region and other regions also increased by 21 (from 28 for Category_V3 to 49 for Category_V3B). This happened because more of the pairs that were outright mis-categorised by Category_V3 are now affected by cross-regional encounters when categorised with Category_V3B. Where previously the categoriser always designated a different region to that determined by STCA, now the pairs are subject to the effect described in section 4.3.1.

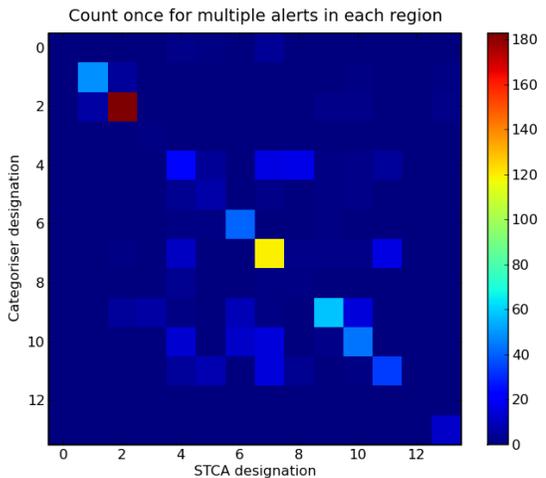
The confusion matrices in figure 4.19 confirm that the new categorisation process is more effective as a reduction in scattering is seen. If the region designations between STCA and Category_V3 were identical we should see a diagonal line plotted from top left to bottom right; although not perfect, the intensity of the diagonal is a marked improvement over that generated by Category_V3 (figure 4.12). A side by side comparison of Category_V3 and Category_V3B assignments can be seen in figure 4.20 (duplicated from figures 4.12(b) and 4.19(b)).



(a) STCA region assignments for alerting pairs vs. Category_V3B region assignments in the dataset.



(b) STCA region assignments for alerting pairs vs. Category_V3B region assignments in the dataset with count restricted to a maximum of 500 to highlight smaller figures.



(c) STCA region assignments for alerting pairs vs. Category_V3B region assignments in the dataset with counts only incremented for each unique result (region combination per pair) across all 57 parameter configurations.

Figure 4.19: Category_V3B discrepancy matrices. Each matrix indicates the number of region assignments for alerting pairs for a particular categoriser designation and the region in which they actually alerted.

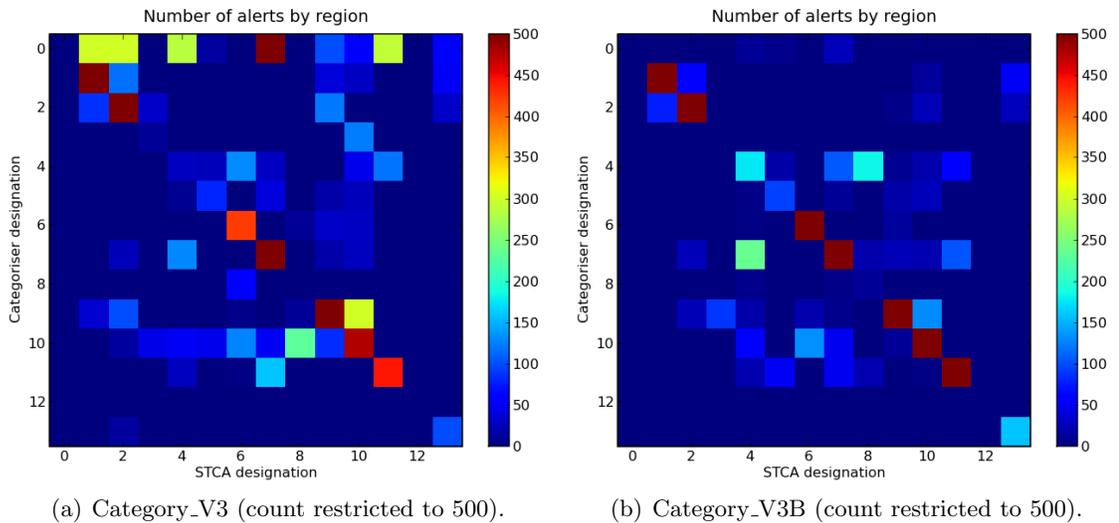


Figure 4.20: Comparison of Category_V3 vs. Category_V3B discrepancy matrices. Each matrix indicates the number of region assignments for alerting pairs for a particular categoriser designation and the region in which they actually alerted.

4.3.4 The region 0's discrepancy

STCA reserves 'region 0' as a special case. Aircraft pairs in conflict within this region are inhibited from alerting. This is necessary around airport runways, for instance, where a large number of aircraft will intentionally be in close proximity; region 0 is used to prevent spurious nuisance alerts. Region 0 has no configurable parameters/rules, as aircraft found inside the boundary simply will not generate an alert.

However, due to the previously mentioned issue with cross-regional aircraft tracks (section 4.3.1), it is possible that in certain situations STCA will alert to a pair that has been classified as region 0 in the optimisation dataset. As STCA has no associated parameters for region 0, the optimiser cannot amend any settings to adjust these alerts. In addition, there is no guarantee that STCA will always classify the pair in the same region, as it is the parameter settings for the surrounding regions, that the aircraft also passed through, that will control if/when an alert is triggered.

In consequence, pairs found to be assigned region 0 in the dataset are ignored by the optimiser. The number of pairs for which this occurs is small and, as can be seen from figures 4.13–4.15, appears not to be significantly detrimental to the optimisation. In addition, correction of the categoriser (Category_V3) to use closest point of approach (CPA) was found to virtually eliminate this problem; compare categoriser region 0 in figures 4.20(a) and 4.20(b) (duplicated from figures 4.12(b) and 4.19(b)).

4.3.5 Discrepancies summary

This section has presented three possible causes of regional parameter dependencies. Due to STCA's use of a sliding-window to confirm filter hits, it is not possible to eradicate all instances of cross-regional aircraft encounter dependencies. However, we have suggested

possible parametrisations for alleviating the effect of ‘incorrect’ dataset labelling and region 0 assignment discrepancies. All subsequent optimisation trials will take these findings into consideration, either by making use of the manually corrected MACC dataset or using the new ‘Category_V3B’ tool to aid dataset categorisation and assign regions based on aircraft Closest Point of Approach. It should be noted however, that the presence of non-regional values in the STCA parametrisations are themselves a continuing source of dependency between regional subsets.

In the next section we investigate whether the estimated-archive algorithm can be protected against noise accumulation.

Algorithm 9 Modified estimated-archive `Main()`

Require: N

Number of iterations

```
1:  $I := \text{initialise}()$             $I$  contains a set of parametrisations, some of which
                                may dominate others (section 3.1.1, page 47)
2:  $A := \emptyset$                  Initialise ‘overall’ archive as an empty set
3: for  $\theta$  in  $I$  do
4:    $\text{update}(A, \theta)$          Update overall archive,  $A$ , with  $\theta$  (algorithm 2, page 47)
5: end for                       End iteration through parametrisations in  $I$ 
6:  $E := A$                        Initilise ‘estimated’ archive,  $E$ , with the contents of overall archive,  $A$ 

7: for  $n := 1$  to  $N$  do
8:    $\theta := \text{select}(E)$        Select a parametrisation,  $\theta$ , from estimated archive,  $E$ 
                                (uniform or uniselect selection, section 3.1.2, page 48)
9:    $\theta' := \text{perturb}(\theta)$   Perturb parent to create a new child,  $\theta'$ ; perturb must ensure
                                that at least 1 parameter from each region is changed
10:   $(TP(\theta'), FP(\theta')) := STCA(\theta')$   Evaluate child’s TP and FP rates on STCA
11:   $\text{update}(A, \theta')$        Update overall archive,  $A$ , with child parametrisation  $\theta'$ 
                                (algorithm 2, page 47)
12:   $E := A$                        Re-initilise estimated archive,  $E$ , with the contents of overall archive,  $A$ 
13:   $\text{updateEstimates}(E, \theta')$   Update estimated archive,  $E$ , with child
                                parametrisation  $\theta'$  (algorithm 8, page 71)
14: end for                       Loop for  $N$  generations
```

4.4 Estimated-archive revisited

Taking into consideration the effects of regional parameter dependencies on the estimated-archive (sections 4.2.1, 4.3), perhaps a more suitable implementation would be to generate an estimated front from scratch *each iteration*, entirely discarding any estimated parametrisations generated at the previous iteration.

It is anticipated that this will help to reduce the accumulation of noise seen when solely recombining with parametrisations from the estimated archive. This should ensure that child parametrisations are selected having considered the best possible combinations of existing regional parameter subsets, without introducing discrepancies by maintaining estimated parametrisations between iterations.

At the start of each iteration the estimated archive is cleared and re-initialised with the contents of the overall-archive. Estimated parametrisations are then produced by recombining a newly evaluated child parametrisation with all previously evaluated parent parametrisations in the regional archives. New members of the estimated-archive are tested for dominance against current members, and any estimated parametrisation found to be entirely dominated by an existing member discarded. Subsequent child parametrisations are selected for perturbation from the freshly populated estimated-archive.

Algorithm 9 outlines the modified estimated-archive procedure more formally. It is very similar to that outlined in algorithm 7 on page 70, with amendments occurring on lines 6 and 12 (highlighted in yellow). At line 1 a set of parametrisations (I) is initialised and

each member evaluated on the STCA system (see section 3.1.1 on page 47 for initialisation options). However, dominance of the members is not determined at this stage. Some of the parametrisations may therefore dominate others. Line 2 initialises the ‘overall’ archive, A , as an empty set. Lines 3 to 5 update the overall-archive with each parametrisation, θ , in set I . At line 6 the estimated-archive, E , is initialised with the contents of the overall-archive, A ; at this point both archives possess the same parametrisations.

The evolutionary optimisation loop is entered at line 7. At line 8 a parent parametrisation, θ , is selected from the *estimated* archive, E , using either uniform or uniselect selection (see section 3.1.2, page 48). The parent parametrisation is then perturbed to create a new child parametrisation, θ' , at line 9 (see section 3.1.3 on page 50 for more on perturbation). Perturbation must ensure that at least one parameter from each regional subset is changed. At line 10 the child parametrisation, θ' , is evaluated on the STCA system. The TP and FP rates for the parametrisation are obtained. At line 11 the overall archive, A , is updated with θ' and at line 12 the estimated archive, E , is *cleared* and reassigned the current contents of the overall-archive, A . At line 13 the estimated archive, E , is updated using the ‘updateEstimates’ method. The algorithm then loops back to line 7 and repeats the process N times.

The `updateEstimates` method described in algorithm 8 (page 71) is used to generate multiple estimated parametrisations by combining each existing member of the estimated archive, E , with new evaluated parametrisation θ' . The estimated archive is updated with each new estimated parametrisation; this means that for every evaluated parametrisation, θ' , archive E will be updated with multiple estimates.

4.4.1 Results of the modified estimated-archive

Figure 4.21 shows the archives’ contents after 12984 iterations on two weeks ($\approx 70,000$ aircraft pairs) MACC July 2007 data (with manually corrected region labellings, see section 4.3.3 page 78). The optimiser has located an actual (evaluated) ROC consisting of 65 points ranging from 5.53% to 65.33% True Positive and 0.00% to 1.64% False Positive (green diamonds). An ‘estimated’ ROC consisting of 82 points, lies from 5.53% to 65.33% TP, 0.00% to 1.61% FP (purple diamonds). The trial was initialised with 100 randomised parametrisations. The NATS manually tuned COP has been marked as a black cross (66.83% TP, 1.33% FP). In this case the NATS COP has not been dominated.

Comparing the evaluated front (green diamonds) to those generated by previous trials (figure 4.22), we see that it has actually performed worse than the original estimated-archive mode (yellow diamonds) for higher TP rates, performing marginally better below $\approx 20\%$ TP and roughly equivalent in the region $\approx 28\%$ to $\approx 38\%$ TP. More significantly, the modified estimated-archive has performed worse than the single-archive optimiser (our base case) for TP rates above $\approx 50\%$. This signifies that protecting against noise accumulation by regular purging of the estimated-archive was perhaps a naive approach.

One possible reason for such poor performance is the reduction in population diversity that results from the purging process. Indeed, compared to multi-archive mode, the estimated and modified estimated-archives *both* suffer from a reduced gene pool. Although

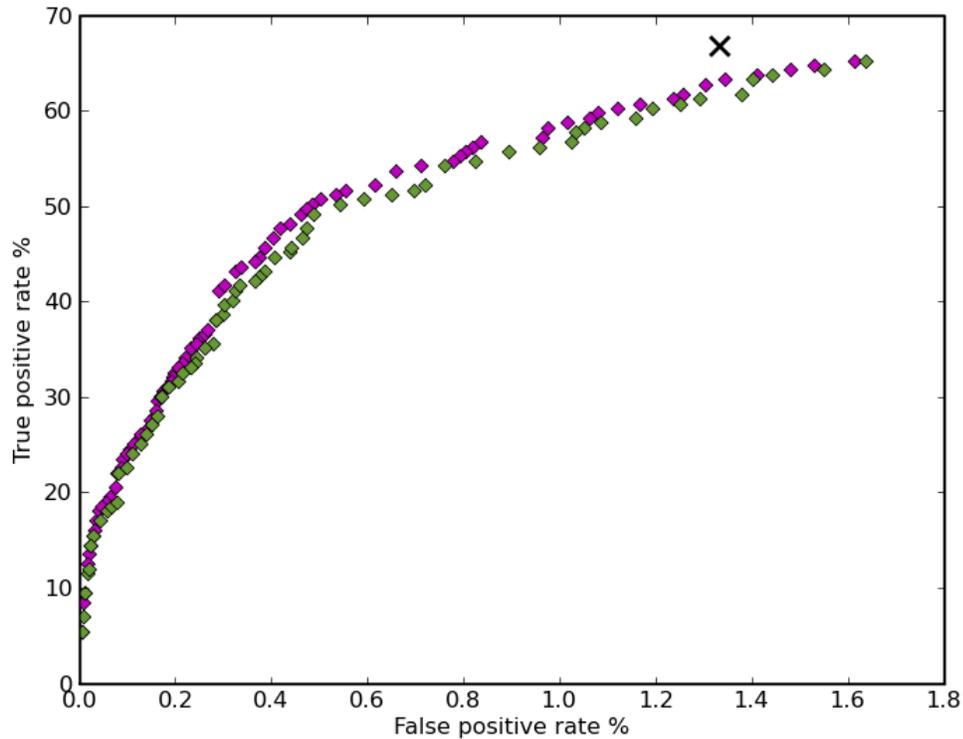


Figure 4.21: Modified estimated-archive optimisation results, corrected MACC dataset (12984 iterations). Green diamonds represent the evaluated ROC, purple diamonds the ‘estimated’ ROC. The NATS COP has been marked as a black cross.

the estimated-archive is conceptually very similar to multi-archive mode, the process of performing domination checks and recombining amongst only estimated parametrisations potentially means that far fewer ‘good’ parameter values are considered than when splicing from multiple, independently maintained, fronts (as with multi-archive mode). For instance, recombination may discard certain parameter values early on, that when combined with later parametrisations might have lead to a better overall TP/FP rate. By maintaining multiple independent archives, multi-archive mode sustains a broader range of values leading to a more diverse number of possible combinations during selection.

This explanation also applies when explaining why single-archive mode betters the modified estimated-archive in some regions of the ROC. Single-archive maintains a varied range of values in the parametrisations, whereas the modified estimated-archive is highly elitist, not only in determination of overall TP/FP rate dominance but also at regional TP/FP rate.

The differences in population diversity between optimisation modes may be summarised as follows:

- The single-archive allows for ‘less than optimal’ regional parameter values, as long as the overall TP/FP rate is not dominated the parametrisation will be accepted into the archive.

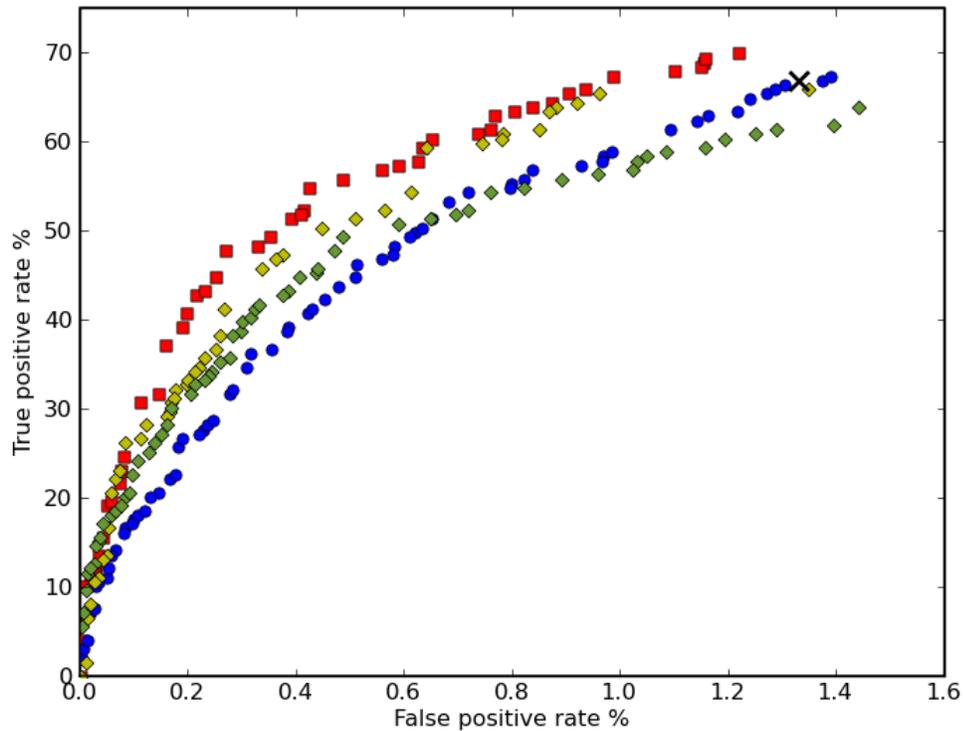


Figure 4.22: Comparison of single, multi, estimated and modified estimated-archive modes after 11900 iterations each on the corrected dataset. Blue circles represent the single archive mode, red squares multi-archive, yellow diamonds the original estimated-archive, green diamonds the modified estimated-archive, the cross indicates the manually tuned operating point.

- The original estimated-archive searches for the ‘best possible’ combination of regional parameter values. This process rejects values that are not optimal in the current set of parametrisations. These values may have been of use in future generations however, they do not survive to propagate and consequently will never be put to the test. The process of maintaining estimates over many generations leads to enough diversity to better single-archive mode (but not the multi-archive optimiser).
- In the modified estimated-archive previous estimates are deleted each iteration and what diversity exists in the ‘overall archive’ is substantially reduced during recombination. This optimisation mode thus performs poorly and fails to dominate single-archive mode across the whole ROC.
- The multi-archive optimiser possesses the benefits of greater population diversity (by maintaining multiple independent archives) and a selection scheme that attempts to ‘splice’ regional parameter values in an intelligent manner. It therefore allows potentially optimal regional values to be matched but not at the expense of diversity. This results in a performance superior to all other modes investigated so far.

4.4.2 Summary

In this section we have investigated a potential solution to the accumulation of noise in the estimated-archive. Trial results have shown that regular purging of solutions in the estimated-archive actually reduces the performance of the optimiser. In examining the differences between parameter selection in the original and modified estimated-archive modes, we speculate that a lack of population diversity causes the optimisation to stall. In attempting to fix the noise accumulation problem in the modified estimated-archive we actually made diversity worse, hence the modified optimiser did not better the results of the original estimated-archive.

Section 4.3 has shown that noise introduced by cross regional aircraft and parameter dependencies is a principal limiting factor in the original estimated-archive conversely, however, the modified estimated-archive suffers most from the less diverse selection scheme inherent in the algorithm itself.

4.5 Summary

In this chapter we have demonstrated how use of multiple archives *can* be used to speed up the optimisation of STCA, reducing the number of iterations necessary for the evolutionary algorithm to generate a Pareto-optimal ROC, and bettering the single-archive optimiser. However, we have highlighted that for the scheme to be most effective regional parameter independence is required.

It is thought that the estimated-archive is particularly susceptible to the introduction of noise caused by parameter dependencies; the accumulative nature of the scheme always selecting *estimated* parents from which to create child generations eventually inhibiting further ROC progression. However, even after applying the types of correction detailed in sections 4.3 and 4.4, the estimated-archive was found not to outperform the multi-archive optimiser. This is likely still due in part to remaining dependencies caused by non-regional parameters and uncorrected dataset labelling discrepancies attributed to cross-regional aircraft encounters. However, the primary cause adversely affecting the *modified* estimated-archive is thought to be a less diverse selection scheme than that employed by the multi-archive and original estimated-archive optimisers, leading to stagnation in the population of parametrisations.

In the next chapter we will describe a method of safely exploring unknown parameter ranges in order to further ROC progression. The algorithm can be augmented into the single-archive, multi-archive or estimated-archive optimisation modes. (See appendix E for stills from a comparison video, further illustrating the progress of various optimisation modes.)

Chapter 5

Aggressive optimisation

The trials presented so far have been labelled ‘conservative’, because perturbation of parameters is restricted to ranges previously used by NATS at the Current Operating Point. Conservative values were determined by analysing all previously used STCA settings (over the past 12 years) for each parameter. This was done as a validation exercise, ensuring that the optimiser could reach (and exceed) the COP without explicit knowledge of it. It also limited potential problems associated with unknown parameter ranges causing STCA system instability, providing an assurance that the optimised system was still operating within the usual parameter ranges. The following section describes an ‘aggressive’ algorithm designed to explore the unknown parameter ranges in a ‘safe’ manner.

5.1 Aggressive optimisation

STCA has a hardcoded range specified for each parameter which were defined during its development. It should be noted that due to the complexity of STCA, there are combinations of parameters, *within* the hardcoded ranges, that can cause the system problems. In addition, due to coupling between the parameters, it is not possible to simply find the safe range for one parameter alone, as its outcome depends upon the value of others in the parametrisation as a whole. Therefore, as a result of the interdependence, individual searches on each parameter cannot be performed; the exploration of unknown ranges must be implemented as part of the optimisation process itself.

The process of determining safe parameter boundaries is inherently tied into the optimisation algorithm; implementing optimisation at the same time as expanding parameter ranges ensures that only relevant boundary modifications are made. That is, there is no point in expanding ranges that do not result in a better optimum, only those ranges restricting relevant search space exploration should be modified.

The basic principle is to increase a parameter’s maximum range until the STCA model becomes unstable (and crashes). For each parameter eight values are tracked (figure 5.1): the conservative or ‘safe’ min/max, the ‘best’/most exploratory values achieved so far, the min/max values that last resulted in an STCA crash, the STCA hardcoded min/max range

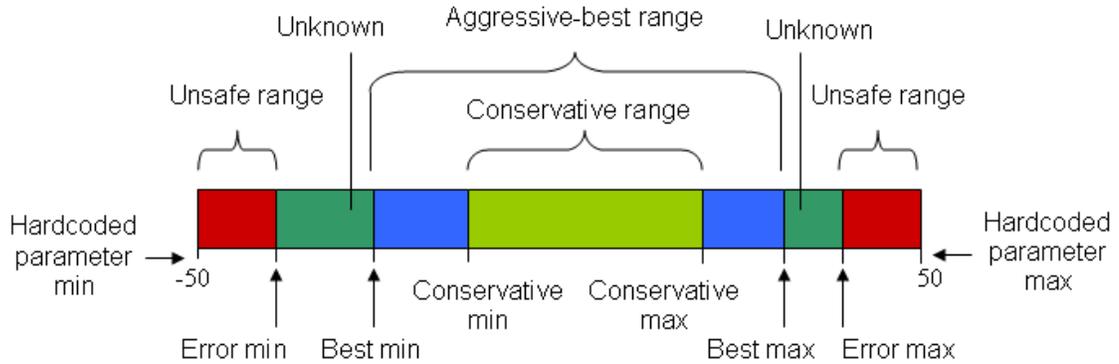


Figure 5.1: Parameter ranges defined for the aggressive optimiser.

as defined in the STCA specification. At initialisation the ‘best’/most exploratory values will equal the conservative or ‘safe’ range; the values that ‘last resulted in an STCA crash’ will be set to the hardcoded range.

Each time a parametrisation is selected for perturbation within the optimiser only one of the parameters selected for mutation may be modified outside of the ‘best’/most exploratory range reached so far; all other parameters are mutated within previously explored ranges. If STCA remains stable the ‘best’ so far value is updated; should a crash occur the ‘error max’ or ‘error min’ value for the parameter is updated.

The actual method of mutation (i.e. random incrementation/decrementation based on Laplacian/Gaussian probability distribution) remains unchanged. Note: this means that just because a parameter may be modified outside of the ‘best’/most exploratory range so far, perturbation does not necessarily guarantee that it will be. This ensures that parameter values closest to bounds are more likely extend the range, whilst those parameters whose optimal value is in a more conservative range leave their boundaries unchanged.

It is trivial to amend the algorithms previously defined for all other optimisation modes to include aggressive parameter perturbation. The required modifications are as follows:

- An additional method ‘updateRanges’ is to be called when parametrisation evaluations complete (see algorithm 10).
- The ‘perturb’ method responsible for parameter value perturbation (based on Gaussian or Laplacian distributions) is permitted to use the ‘aggressive-best’ range defined for a given parameter (which is initially equal to the ‘conservative’ range). Values may be perturbed outside of the ‘aggressive-best’ range once per parametrisation, after which all subsequent perturbations performed on the parametrisation must lie *within* the ‘aggressive-best’ range.

Algorithm 10 $\text{updateRanges}(\theta'_k)$

Require: $completed$ Boolean indicating whether STCA completed evaluation
Require: $eMin$ Lowest value that last caused STCA to fail evaluation for parameter θ_k
('error min')
Require: $eMax$ Highest value that last caused STCA to fail evaluation for parameter θ_k
('error max')
Require: $bMin$ 'Best' minimum value found so far for parameter θ_k
('best min')
Require: $bMax$ 'Best' maximum value found so far for parameter θ_k
('best max')

```
1: if  $(\theta'_k > eMin) \ \&\ (\theta'_k < bMin)$  then
2:   if  $\neg completed$  then
3:      $eMin := \theta'_k$  Expand the min 'unsafe' range
4:   else
5:      $bMin := \theta'_k$  Expand the 'aggressive-best' range
6:   end if
7: else if  $(\theta'_k < eMax) \ \&\ (\theta'_k > bMax)$  then
8:   if  $\neg completed$  then
9:      $eMax := \theta'_k$  Expand the max 'unsafe' range
10:  else
11:     $bMax := \theta'_k$  Expand the 'aggressive-best' range
12:  end if
13: end if
```

Algorithm 10 describes how parameter ranges are updated after each parametrisation's evaluation. If the perturbed parameter value θ'_k is greater than $eMin$ and less than $bMin$ (line 1), and STCA failed to evaluate parametrisation θ' (line 2) then $eMin$ is assigned the value of θ'_k (line 3). If STCA did complete evaluation of parametrisation θ' then $bMin$ is assigned the value of θ'_k (line 5).

If the perturbed parameter value θ'_k is less than $eMax$ and greater than $bMax$ (line 7), and STCA failed to evaluate parametrisation θ' (line 8) then $eMax$ is assigned the value of θ'_k (line 9). If STCA did complete evaluation of parametrisation θ' then $bMax$ is assigned the value of θ'_k (line 11).

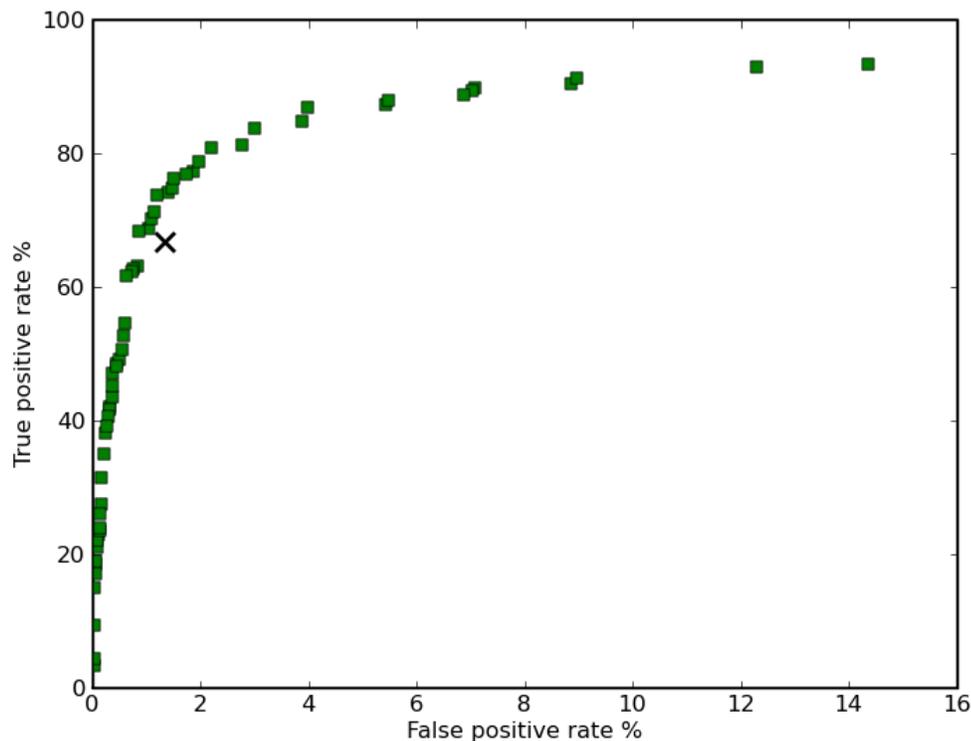


Figure 5.2: Multi-archive, aggressive optimisation results evaluated on the MACC dataset with corrected regions (14402 iterations). The NATS COP has been marked as a black cross.

5.1.1 Results of the aggressive optimiser

Figure 5.2 shows the multi-archive, aggressive, ‘overall’ archive’s contents after 14402 iterations on two weeks ($\approx 70,000$ aircraft pairs) MACC July 2007 data (with manually corrected region labellings, see section 4.3.3 page 78). The optimiser has located an ROC consisting of 59 points ranging from 3.52% to 93.47% True Positive and 0.00% to 14.35% False Positive. The trial was initialised with 100 randomised parametrisations. The NATS manually tuned Current Operating Point (COP) has been marked as a black cross (66.83% TP, 1.33% FP).

Figure 5.3 compares the aggressive optimisation run (green squares) to a previous ‘conservative’ trial on the same data (red squares) after 11900 iterations. At low TP and FP rates the the conservative and aggressive fronts virtually coincide. The additional freedom allowed to the parameters in the aggressive case permits new solutions at higher TP and FP rates to be discovered, resulting in a more complete picture of the Pareto optimal curve. The COP is now observed to be close to the corner or ‘knee’ of the ROC and the optimiser has located operating points with performance well beyond that of the COP. For approximately the same tolerated FP rate the optimiser has located a parametrisation $\approx 7.5\%$ better in TP rate than the COP, for approximately the same tolerated TP rate the optimiser has located a parametrisation $\approx 0.5\%$ better in FP rate than the COP.

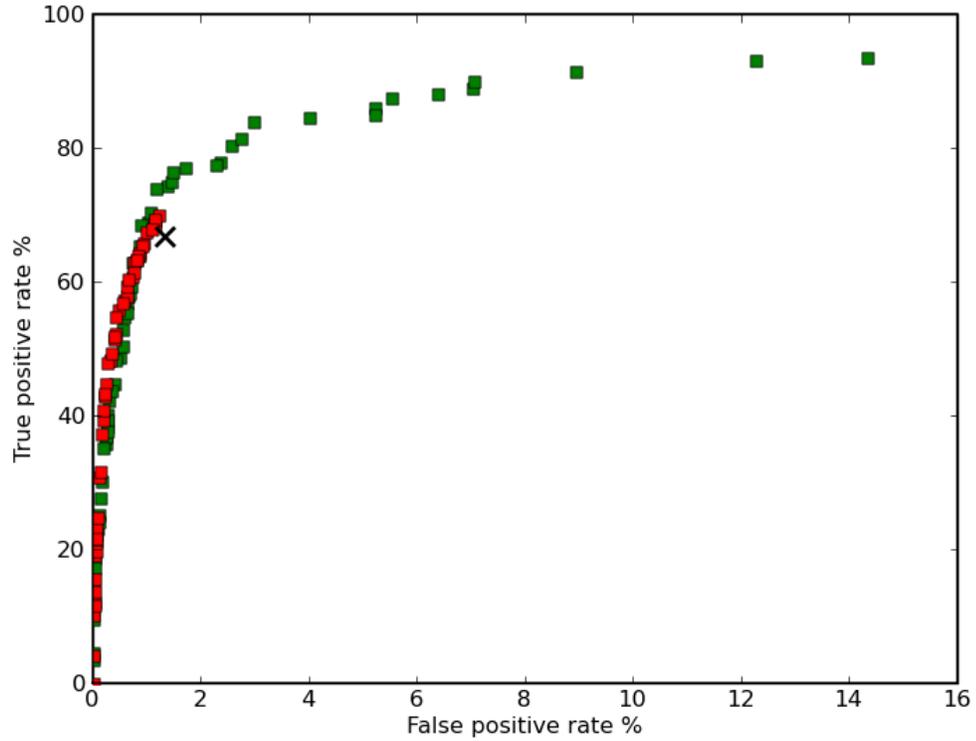
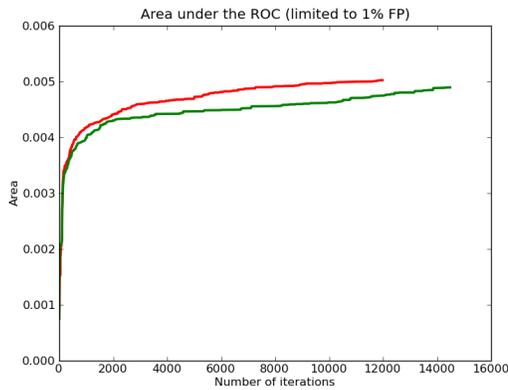
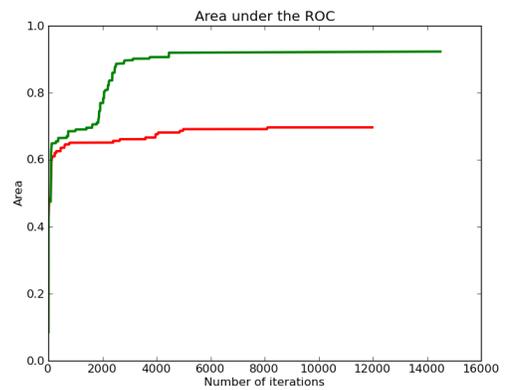


Figure 5.3: Comparison of Multi-archive conservative (green) and aggressive (red) optimisation results on the MACC dataset with corrected regions (11900 iterations).



(a) Area under ROC, limited to 1% FP.



(b) Area under ROC, full.

Figure 5.4: Comparison of convergence between Multi-archive conservative and aggressive optimisation modes using the area under ROC.

Figure 5.4 compares plots of the area under the ROC as a measure of convergence speed. The aggressive optimisation is again shown in green, conservative in red. In figure 5.4(a) the area has been limited to below 1% FP rate, so as to highlight smaller changes. Figure 5.4(b) shows the area under the ROC in full. It can be seen that at low FP rates the aggressive archive’s convergence is just behind that of the conservative archive (figure 5.4(a)), this slower progression is likely due to perturbations/development in other higher TP/FP areas of the aggressive ROC (that the conservative archive does not cover). The vast difference seen when examining the area under the ROC in full (figure 5.4(b)) suggests that this is the case.

Use of the aggressive optimisation mode has little or no impact on optimisation speed when compared to the equivalent conservative trial. That is, although there are some wasted perturbations due to evaluation failures (see figure 5.5), these are more than made up for by the overall improvement in parametrisation quality. Thus, the aggressive mode can produce a better quality ROC without having a negative impact on the time taken to generate it. Additionally, by analysing those parameter settings that caused STCA to become unstable, the optimiser has highlighted several instances of incorrectly defined variable ranges in the STCA specification.

We can see from figure 5.5 that the evaluation failure rate for this trial is roughly 1 in 10; a total of 1461 failures occurred. However, of those that we examined it was discovered that most were caused by incorrectly specified hardcoded ranges (either in the STCA specification or due to data entry errors when compiling the parameter map).

Figures 5.6 and 5.7 illustrate the extent to which some parameter ranges have been extended. The charts have been normalised to each parameter’s min/max conservative range (marked in green). The yellow points represent how far the optimiser has extended the aggressive-best range into unknown parameter space, blue points mark each parameter’s current value in the given parametrisation. Figure 5.6(c) shows the normalised ranges for the bottom-left parametrisation in figure 5.2, figure 5.6(a) the parametrisation which directly dominates the NATS COP in figure 5.2 (the corner or ‘knee’ of the ROC; 68.84% TP, 1.03% FP) and figure 5.6(b) the top-right parametrisation in figure 5.2. Figure 5.6(d) shows the normalised ranges for the NATS COP itself, note how all parameter values for this plot lie *within* the conservative range.

Figure 5.7 is a duplicate of 5.6(a), however, it has not been scaled to highlight smaller values and thus shows the true extent to which some ranges have been extended. Note the significant number of parameter values that lie outside of the conservative range (marked in blue).

5.2 Summary

This chapter has presented a method of incrementally exploring ‘unknown’ regions of STCA parameter space in a ‘safe’ and efficient manner. The concept is similar to that of Tabu search [45] in that we are effectively keeping track of ineffective parameter values. In doing so, the method explores parameter space as a hyper-rectangle, preventing the optimiser from

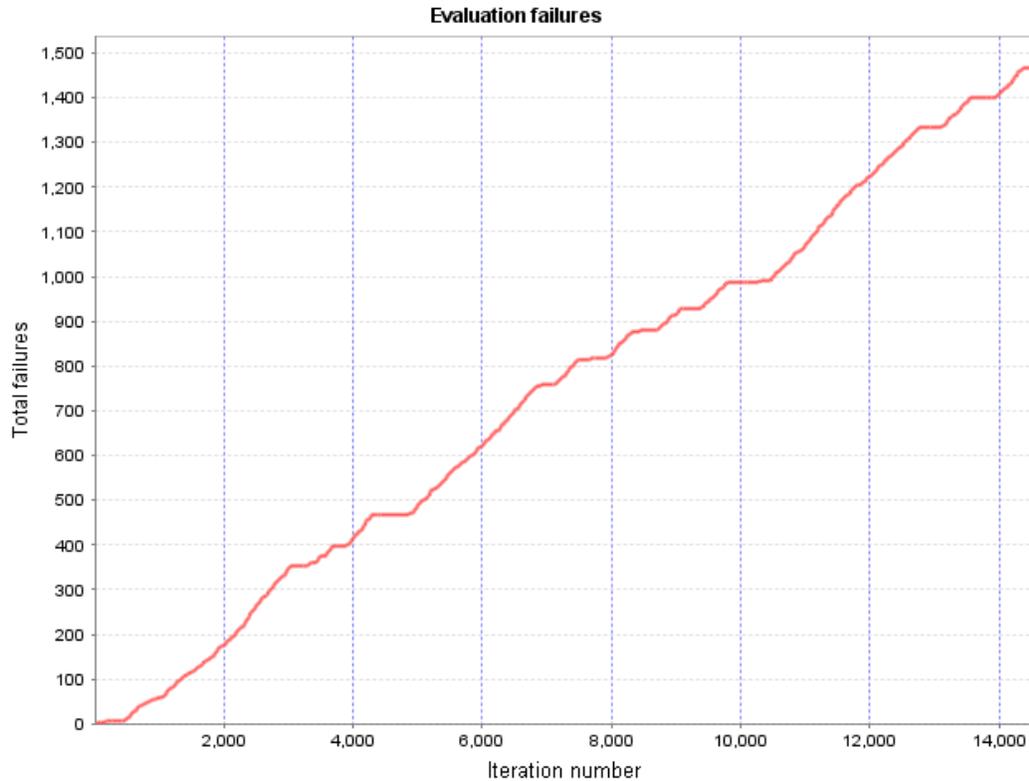
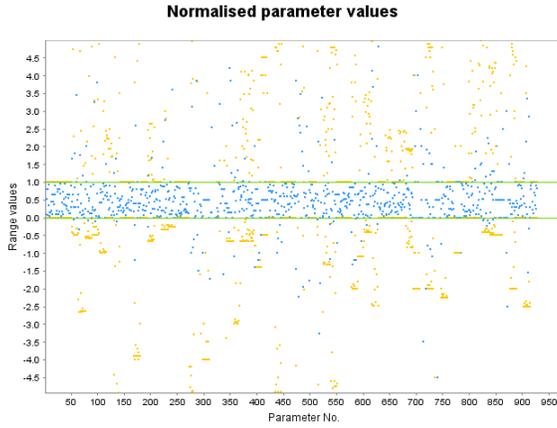


Figure 5.5: Cumulative number of objective evaluation failures during Multi-archive aggressive optimisation on the MACC dataset, showing that the number of failures does not markedly increase as the conservative ranges are expanded (14402 iterations).

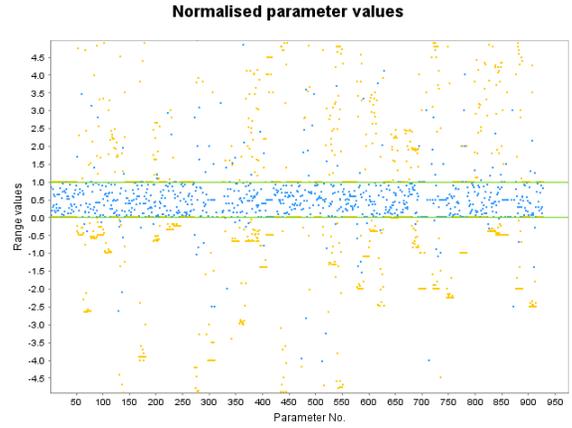
repeating ‘mistakes’ and unnecessarily wasting evaluations.

The results have shown that a more complete picture of the Pareto-optimal ROC may be obtained without increasing the number of necessary iterations. Additionally, analysis of failed STCA parametrisations has uncovered previously unknown STCA system/specification errors.

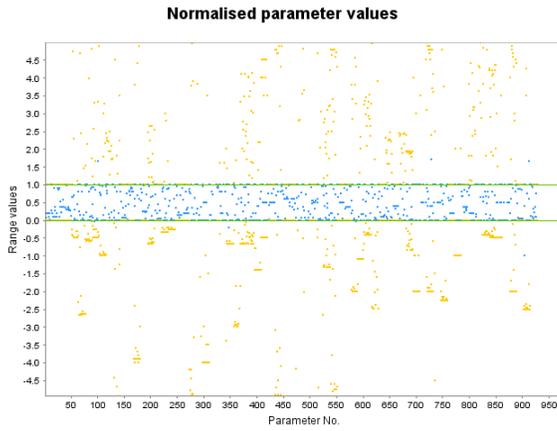
‘Multi-archive’ mode with aggressive perturbation is by far the most effective means of optimisation that we have investigated.



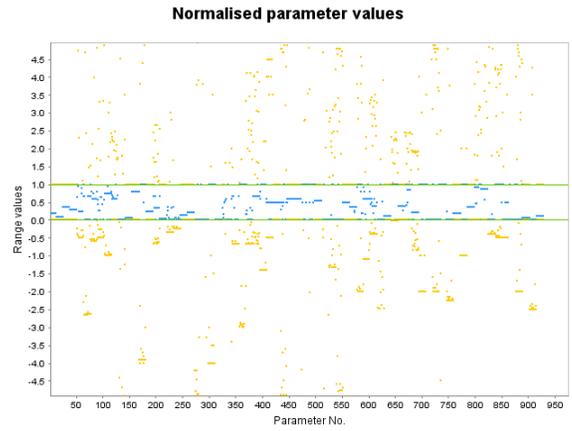
(a) Parametrisation that close to the corner of the ROC.



(b) Parametrisation that lies at the top-right of the ROC.



(c) Parametrisation that lies at the bottom-left of the ROC.



(d) NATS COP.

Figure 5.6: Parameter ranges plot normalised to the conservative range. Plots are scaled to $[-5, +5]$. Green represents the conservative min/max for each parameter, yellow the aggressive-best min/max range, and blue each parameter's current value in the given parametrization. Parameters are listed by one parameter for all regions followed by the next parameter for all regions.

Normalised parameter values

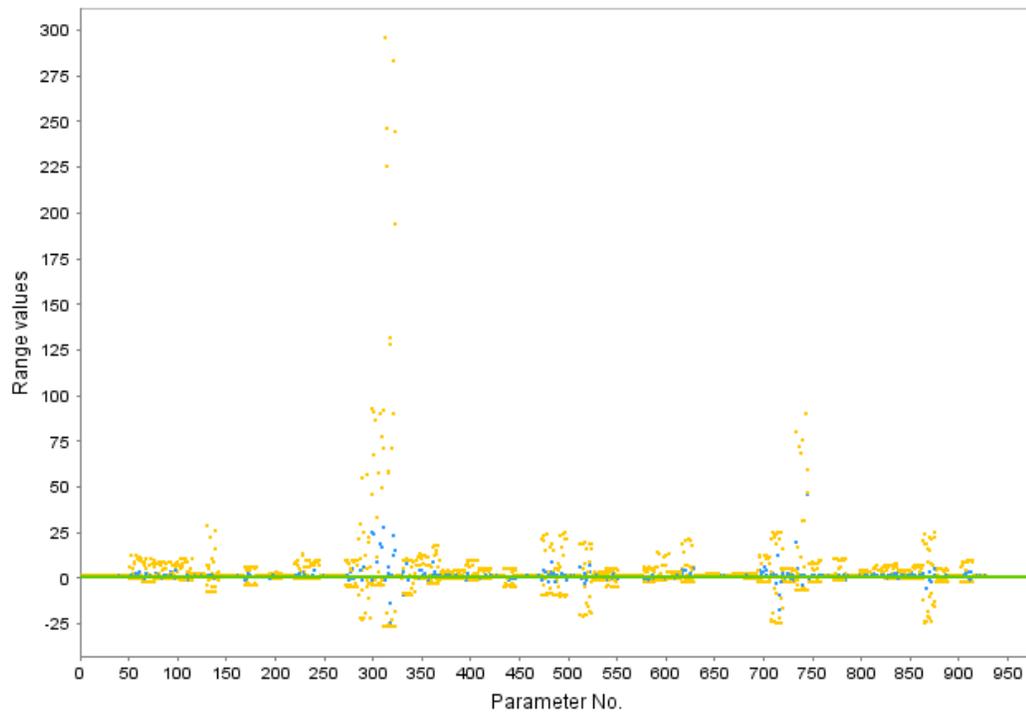


Figure 5.7: Parametrisation that lies at the corner of the ROC: Parameter ranges plot normalised to the conservative range. The plot has not been scaled (full ranges shown). Green represents the conservative min/max for each parameter, yellow the aggressive-best min/max range, and blue each parameter's current value in the given parametrisation.

Chapter 6

Single-parent optimisation and non-regional parameters

In this chapter we investigate whether further modification of the parental selection scheme can improve archive convergence. So far three main selection schemes have been investigated; uniform, uniselect and global uniselect (depending on optimiser mode used, see sections 3.1.2, page 48 and 4.1.1, page 61). However, all of these schemes are a variation on the same theme, allowing multiple parents to be selected at random, possibly with some additional bias imposed.

The work of [72], [73] and [61] seems to suggest that a better scheme may in fact be to keep the same parametrisation throughout all iterations, applying perturbations to this parametrisation alone and evaluating the resulting variations. We discuss the effectiveness of applying such a scheme to optimisation of STCA.

6.1 Single-parent optimisation

As is discussed in [72], the ‘state’ of an optimisation is traditionally represented in one of two ways; one can regard the archived set of non-dominating parametrisations as the state (as in all our previous trials), or alternatively the state may consist of a *single* parametrisation. In the former case one might expect better avoidance of local optima since perturbations are made across the entire estimated Pareto front, reducing the likelihood that the optimisation will become ‘stuck’ at a particular parametrisation. Conversely however, it is suggested by [72] that repeated perturbation of a single parametrisation might be more efficient because it can rapidly reach the vicinity of the true Pareto front, before filling out the full extents of the curve. A set-based state is slower in this respect because effectively each member must be perturbed towards the front.

This concept has been illustrated in figure 6.1. In this contrived example, in which as usual we seek to minimise the FPR and maximise the TPR, we assume that at each time-step identical perturbations are made to the parametrisations selected from archive *A* or *B*. It is only the archive member selected that differs between the two diagrams. At each time-step newly dominated members, marked for removal from the archive, are highlighted in pink with

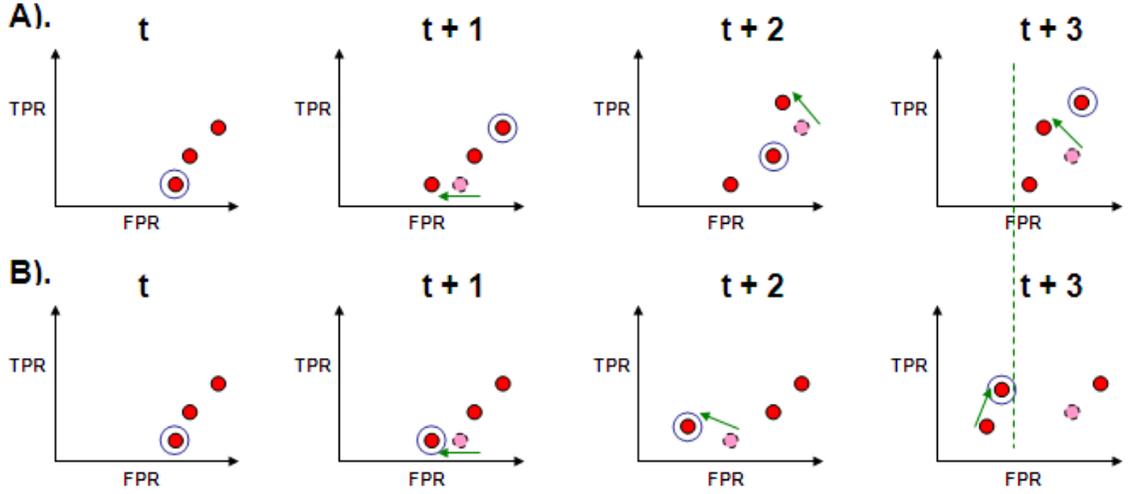


Figure 6.1: Illustration of multiple (A) vs single (B) parent selection. See text for full details.

a dotted border. Members selected for perturbation in the next time-step are highlighted with a blue circle. The direction of movement in each time-step is shown with a green arrow.

At time t , the contents of archives A and B are identical. A tracks the progress of a set-based state, B the progress of a single parametrisation state. At time $t + 1$ the archive contents are still identical as the same parametrisation was selected for perturbation in both A and B at time t . At time $t + 2$ the archives differ, A lies further behind B because the top-right parametrisation was selected at time $t + 1$ instead of the bottom-left. Similarly, at time-step $t + 2$ the central parametrisation is selected in A instead of the bottom left. By time $t + 3$ we can see that the net effect of selecting the same parametrisation each iteration is that archive B has more rapidly located the vicinity of the Pareto front and lies ahead of archive A in all but one member.

6.2 The single-parent, single-archive optimiser

Here we apply the single parametrisation state scheme to the optimiser in single-archive mode (see chapter 3, page 45 for details of the single-archive optimiser). The optimisation procedure shown in algorithm 11 remains largely unchanged from that outlined in algorithm 1 (page 47), except that the `select(A)` method has been replaced in the main optimisation loop.

Referring to algorithm 11 we can see that at line 1 archive A is initialised with a set of randomised parametrisations. From these, a single initial parametrisation, θ , is selected (line 2). At line 3 the main optimisation loop is entered. The initial ‘parent’ parametrisation, θ , is perturbed (line 4) and evaluated on the STCA system (line 5). If the newly evaluated child parametrisation, θ' , is found *not* to be dominated by any current members of archive A , then it is added to the archive and members that it dominates removed (line 6, see algorithm 2, page 47). At line 7 ‘selection’ has been replaced with an assignment, ensuring that the same parametrisation is perturbed throughout all iterations. This guarantees that if changes to the original parent, θ , result in an ‘improvement’ (e.g. that θ' is not dominated by any element

Algorithm 11 Single-parent, single-archive Main()

Require: N Number of iterations

- 1: $A := \text{initialise}()$ Initialise archive A (section 3.1.1, page 47)
- 2: $\theta := \text{select}(A)$ Select a parent, θ , to perturb (section 3.1.2, page 48)
- 3: **for** $n := 1$ to N **do**
- 4: $\theta' := \text{perturb}(\theta)$ Perturb parent to create a new child, θ' (section 3.1.3, page 50)
- 5: $(TP(\theta'), FP(\theta')) := STCA(\theta')$ Evaluate child's TP and FP rates
- 6: **if** $\text{update}(A, \theta')$ **then**
- 7: $\theta := \theta'$ Same parent parametrisation is kept throughout all iterations
- 8: **end if**
- 9: **end for** Loop for N generations

of A), they are kept in the subsequent generation, otherwise the modifications are discarded (and θ remains unchanged).

It should be noted that at each generation more than one child solution may be generated in a $(1 + \lambda)$ -evolutionary strategy, where λ children are generated from the current parent, the best of these becoming the new parent. However, for the purposes of our trials only one child was generated each iteration – a $(1 + 1)$ -evolutionary strategy.

6.2.1 Results of the single-parent, single-archive optimiser

Figure 6.2 shows the archive's contents after 18464 iterations on two weeks ($\approx 70,000$ aircraft pairs) MACC July 2007 data (with manually corrected region labellings, see section 4.3.3 page 78). The optimiser has located an ROC consisting of 66 points ranging from 3.02% to 65.83% True Positive and 0.00% to 1.56% False Positive (orange circles). The trial was initialised with a single randomised parametrisation. The NATS manually tuned Current Operating Point (COP) has been marked as a black cross (66.83% TP, 1.33% FP). In this case the NATS COP has not been dominated. However, the ROC has reached the vicinity of the COP in that there are points non-dominating in respect to it, although it dominates solutions with higher FPR.

Figure 6.3(a) compares single-parent (orange) and multi-parent (blue) modes after 13722 iterations. Comparison plots of the area under the ROC (limited to 1% FP for clarity) vs iteration number are included in figure 6.3(b). From these we can see little difference in either speed of progression or the quality of the resulting estimated Pareto front. The single-parent optimiser does seem to be marginally ahead in terms of progression however, progress itself is far more erratic appearing to stall between ≈ 1000 and ≈ 2000 iterations, and again between ≈ 2500 and ≈ 5000 iterations. This could well be caused by local optima causing the single-parent optimiser to become temporarily 'stuck'.

The single-parent optimiser archive does just dominate that of the multi-parent optimiser between 20% and 45% TP, and again between 54% and 65% TP. However, using the corrected dataset, the multi-parent, single-archive optimiser has managed to locate and match the NATS operating point and includes a higher number of parametrisations from 60% TP upwards.

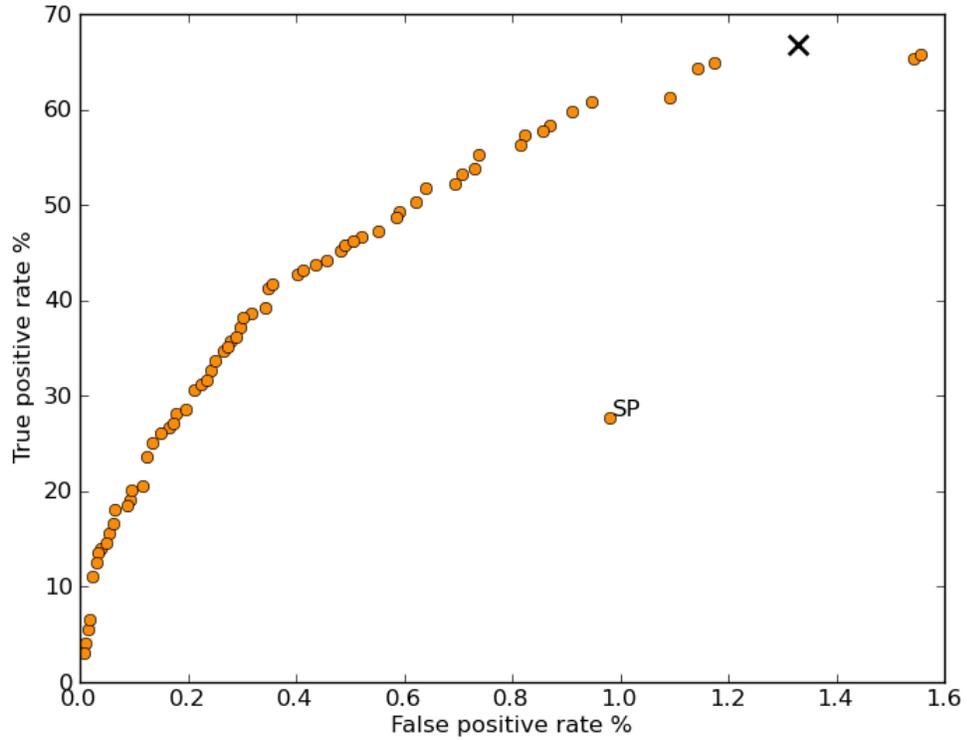
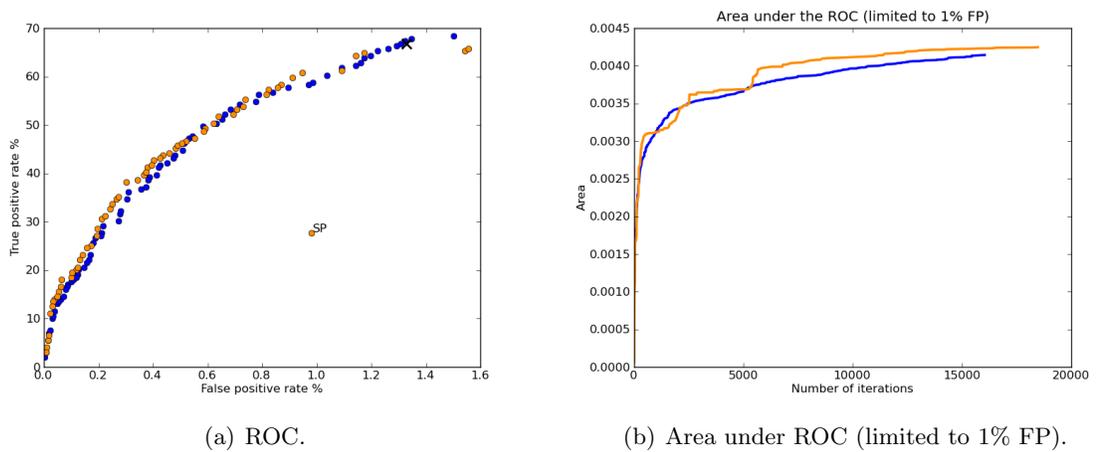


Figure 6.2: Single-parent, single-archive optimisation results, MACC dataset (18464 iterations) with corrected regions. The NATS COP has been marked as a black cross. The (randomised) starting point for the single-parent run has been labelled ‘SP’.



(a) ROC.

(b) Area under ROC (limited to 1% FP).

Figure 6.3: Comparison of single-parent (orange) vs multi-parent (blue), single-archive optimisation results, MACC dataset (13722 iterations) with corrected regions.

6.2.2 Summary

From initial results there appears to be no clear advantage in choosing single-parent over multi-parent optimisation. Overall convergence of the single-parent ROC seems roughly equivalent to that of the multi-parent and there is little benefit in terms of the speed of convergence, as progression appears more erratic. It is possible that the single-parent mode suffers by getting ‘stuck’ at a particular parametrisation, whereas multi-parent mode avoids this hazard by allowing selection of alternative parents.

In the next section we apply the single-parent scheme to our multi-archive optimisation approach.

6.3 The single-parent, multi-archive optimiser

Here we apply the single parametrisation state scheme to multi-archive mode (see chapter 4, page 56 for details of the multi-archive optimiser). We modify our original multi-archive algorithm (algorithm 4, page 60) by removing the `select({Ai}, A)` method from the main loop and instead keep track of a separate ‘single-parent’ for each of the regional archives/regional parameter subsets. Non-regional parameters are handled by randomly (uniformly) selecting one of the non-regional subsets belonging to a contributing regional archive’s ‘single-parent’. A composite parametrisation is then ‘spliced’ together from these parental subsets to form a complete set, which is then perturbed and evaluated.

Algorithm 12 explicitly outlines the single-parent, multi-archive procedure. Lines 1 to 12 are identical to the original multi-archive code (algorithm 4, page 60) and concern only the initial population of the ‘overall’ and ‘regional’ archives with (usually randomised) parametrisations. At line 13 $\{\lambda_i\}$ is initialised as an empty set. This variable is used to track non-regional parameters that belong to contributing regional parameter subsets (one for each STCA region). At line 14 a single initial parametrisation, θ , is selected by splicing regional parameter subsets from selected members of the regional archives (see algorithm 13). $\{\lambda_i\}$ is also populated as a consequence of this.

The evolutionary optimisation loop is entered at line 15. The initial parent parametrisation, θ , is perturbed to create a new child parametrisation, θ' (line 16, see section 3.1.3 on page 50 for more on perturbation). Perturbation must ensure that at least one parameter from each regional subset is changed so that STCA evaluation of the parametrisation, θ' , in that region is not wasted. At line 17 the child parametrisation, θ' , is evaluated on the STCA system. The overall TP and FP rates for the parametrisation are obtained, along with the associated regional TP and FP rates. At line 18 the ‘overall’ archive, A , is updated with θ' using the parametrisations overall TP and FP rates. At line 19 an empty parametrisation, σ , is created. This will be used to ‘splice’ together a new ‘single-parent’, dependent on regional archive updates.

At line 21 a subset of parameters, ϕ , corresponding to region i , is extracted from θ' . The regional parameters subset, ϕ , is then updated to the corresponding regional archive A_i at line 22. Dominance is determined based on the subset’s regional TP and FP rates,

Algorithm 12 Single-parent, multi-archive **Main()**

Require: N Number of iterations
Require: R Number of STCA parameter regions

1: $I := \text{initialise}()$ I contains a set of parametrisations, some of which may dominate others (section 3.1.1, page 47)
2: $A := \emptyset$ Initialise ‘overall’ archive as an empty set
3: **for** $i := 1$ to R **do**
4: $A_i := \emptyset$ Initialise each regional archive as an empty set
5: **end for**
6: **for** θ in I **do**
7: $\text{update}(A, \theta)$ Update ‘overall’ archive, A , with θ (algorithm 2, page 47)
8: **for** $i := 1$ to R **do**
9: $\phi := \text{extractRegion}(\theta, i)$ Extract parameter subset, ϕ , corresponding to region i from parametrisation θ
10: $\text{update}(A_i, \phi)$ Update regional archive, A_i , with ϕ (algorithm 2, page 47)
11: **end for** End update of regional archives with current member, θ , of I
12: **end for** End iteration through parametrisations in I

13: $\{\lambda_i\} := \emptyset$ Initialise set of parents’ non-regional parameters as an empty set
14: $\theta := \text{singleParentCompositeSelect}(\{A_i\}, \{\lambda_i\})$ Select an initial parent, θ , to perturb; populate $\{\lambda_i\}$ (algorithm 13)

15: **for** $n := 1$ to N **do**
16: $\theta' := \text{peturb}(\theta)$ Perturb parent to create a new child, θ' ; **peturb** must ensure that at least 1 parameter from each region is changed
17: $(TP(\theta'), FP(\theta')) := \text{STCA}(\theta')$ Evaluate child’s TP and FP rates on STCA
18: $\text{update}(A, \theta')$ Update ‘overall’ archive, A , with child parametrisation θ' (algorithm 2, page 47)
19: $\sigma := \emptyset$ Initialise temporary parametrisation, σ , as an empty set
20: **for** $i := 1$ to R **do**
21: $\phi := \text{extractRegion}(\theta', i)$ Extract parameter subset, ϕ , corresponding to region i from perturbed parametrisation θ'
22: **if** $\text{update}(A_i, \phi)$ **then**
23: $\lambda_i := \text{nonRegionalParameters}(\theta')$ Extract the ‘non-regional’ parameters from parametrisation θ' and assign them to λ_i
24: **else**
25: $\phi := \text{extractRegion}(\theta, i)$ Extract parameter subset, ϕ , corresponding to region i from original parametrisation θ
26: **end if**
27: $\sigma := \text{append}(\sigma, \phi)$ Append the selected regional parameter subset, ϕ , to composite parametrisation σ
28: **end for** End update of regional archives with child parametrisation, θ'
29: $i := \text{randint}(1, R)$ Select a random index, i , between 1 and the number of STCA parameter regions
30: $\theta := \text{append}(\sigma, \lambda_i)$ Append randomly selected non-regional parameters λ_i to temporary parametrisation, σ , and assign the resulting parameter set to θ (the new parent to be perturbed)
31: **end for** Loop for N generations

6.3.1 Results of the single-parent, multi-archive optimiser

Figure 6.4 shows the archive’s contents after 13529 iterations on two weeks ($\approx 70,000$ aircraft pairs) MACC July 2007 data (with manually corrected region labellings, see section 4.3.3 page 78). The optimiser has located an ROC consisting of 36 points ranging from 8.04% to 46.73% True Positive and 0.01% to 1.16% False Positive (orange squares). The trial was initialised with a single randomised parametrisation. The NATS manually tuned Current Operating Point (66.83% TP, 1.33% FP) has not been dominated. Indeed, the ROC appears stunted, in terms of progression this is the worst performing algorithm examined thus far.

Figure 6.5(a) compares single-parent (orange) and multi-parent (red) modes after 11900 iterations. Comparison plots of area under ROC (limited to 1% FP for clarity) vs iteration number are included in figure 6.5(b). These highlight just how far behind the single-parent, multi-archive front really is; the two ROCS in figure 6.5(a) only briefly overlap between $\approx 8\%$ and $\approx 14\%$ TP. Figure 6.5(b) again indicates that the single-parent scheme suffers from erratic progression (as the optimiser becomes temporarily ‘stuck’). Indeed, after only ≈ 2000 iterations progression of the single-parent, multi-archive optimiser slows dramatically and appears to stall almost completely.

For completeness figure 6.6 compares single-parent/single-archive (orange circles), single-parent/multi-archive (orange-squares), multi-parent/single-archive (blue circles) and multi-parent/multi-archive (red squares) against one another (after 11900 iterations). From this we can see that the single-parent, multi-archive optimiser (orange squares) is actually generating parametrisations roughly equivalent to (and in some cases better than) the multi-parent, single-archive (blue circles) optimiser for TP rates below $\approx 35\%$, however above $\approx 35\%$ the front becomes severely stunted. This figure also highlights how much further ahead (more converged) the multi-parent, multi-archive (red squares) optimiser is compared to the three other optimisation modes illustrated.

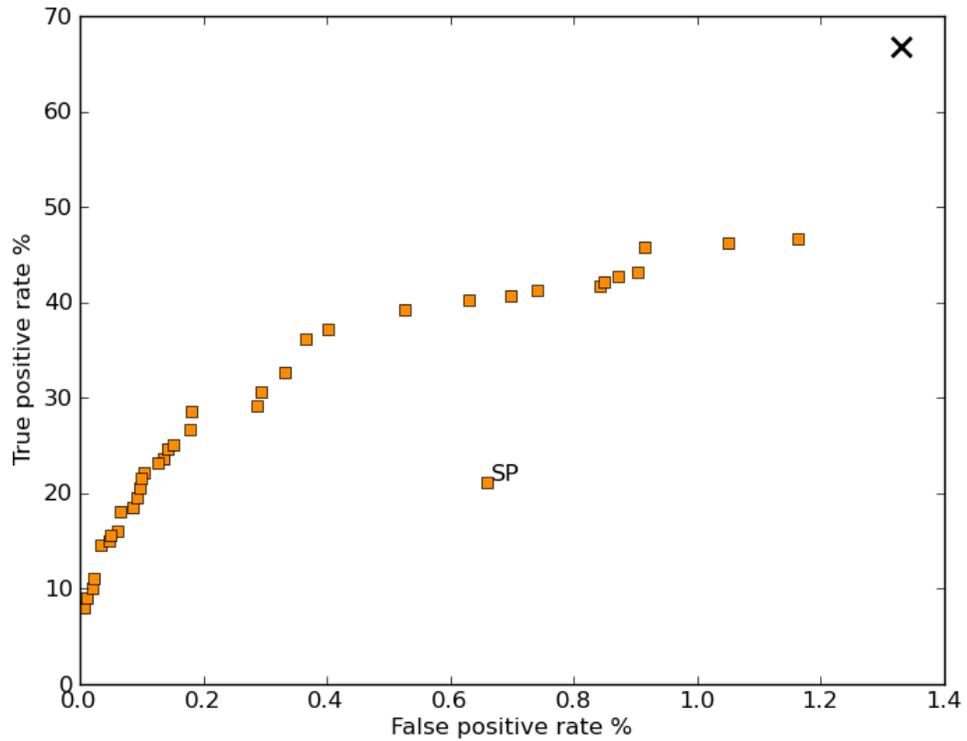
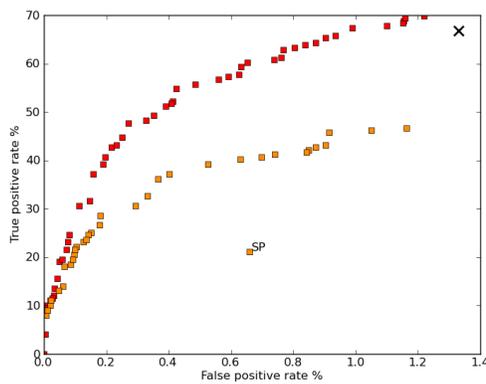
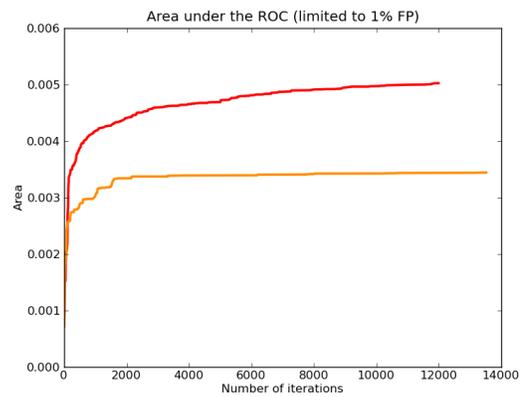


Figure 6.4: Single-parent, multi-archive optimisation results, MACC dataset (13529 iterations) with corrected regions. The NATS COP has been marked as a black cross. The (randomised) starting point for the single-parent run has been labelled ‘SP’.



(a) ROC.



(b) Area under ROC (limited to 1% FP).

Figure 6.5: Comparison of single-parent (orange) vs multi-parent (red), multi-archive optimisation results, MACC dataset (11900 iterations) with corrected regions.

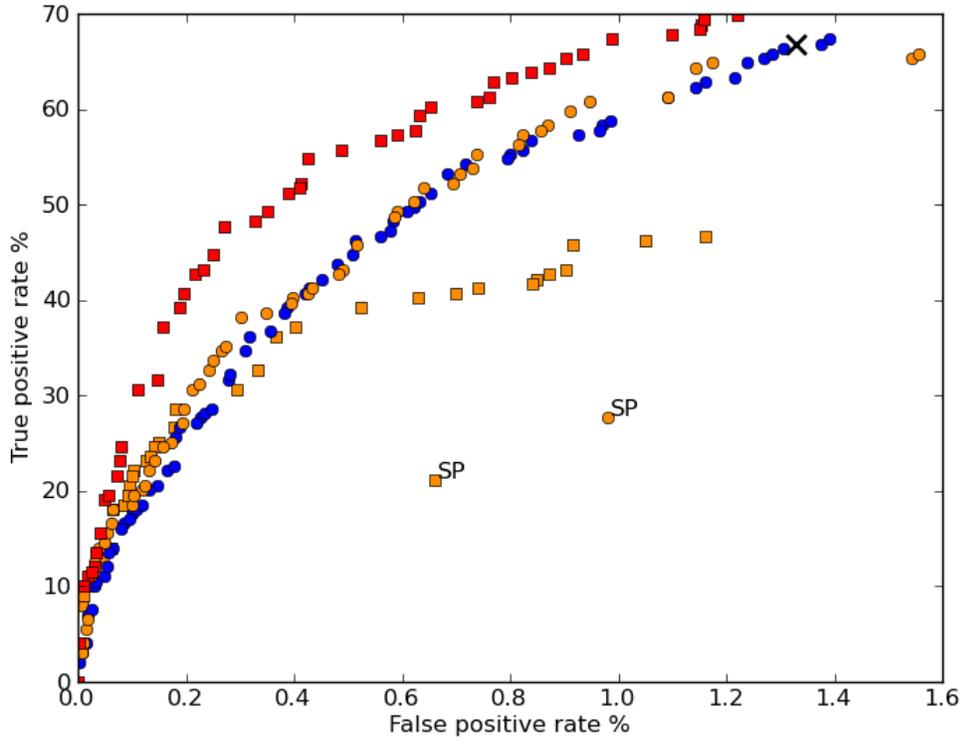


Figure 6.6: Comparison of single (orange) vs multi-parent (blue and red), single (circles) vs multi-archive (squares) optimisation results, MACC dataset (11900 iterations) with corrected regions. The (randomised) starting points for single-parent runs have been labelled ‘SP’.

6.3.2 Summary

Contrary to what is suggested in [73] it appears that, when applied to the optimisation of STCA, there is no advantage in choosing a single-parent scheme over multi-parent selection. In the single-archive case overall ROC convergence seems roughly equivalent, and to multi-archive optimisation single-parent selection appears to be severely detrimental. In both cases progression appears erratic in comparison to multi-parent modes. This is likely caused by the single-parent mode getting ‘stuck’ at a particular parametrisation. It is possible that the particular structure of the ‘DTLZ’ problems used by [73] made single-parent the more attractive selection method, however in our trials it was found to be ineffective.

We remark that single-parent mode cannot be used with the estimated-archive; as the estimated-archive requires the generation of multiple estimated parametrisations (from which a member is selected) they are incompatible concepts and cannot be used in unison.

6.4 Discussion of efficacy 2

In section 4.3 we discussed the flawed assumption that disabling all non-regional parameters from optimisation would lead to regional parameter independence. However, we acknowledged that non-regional parameter dependences are probably a contributing factor to noise. In the following section we will test this hypothesis and investigate whether disabling optimisation of non-regional parameters results in an improvement in the convergence of single and multi-parent optimisation modes.

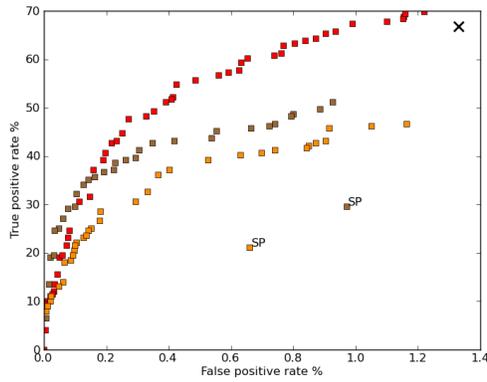
6.4.1 Non-regional parameter dependencies

As stated in section 4.3; if all non-regional parameters were disabled for optimisation (i.e. were set to fixed values in all parametrisations), then one could presume that the regional parameters are completely independent of one another. We have already investigated dependencies resulting from cross-regional aircraft pairs and region-labelling discrepancies, however, in this next section we will investigate what effect disabling optimisation of non-regional parameters has on ROC convergence. We expect that the regional parameters are the greatest source of dependency. Ignoring what we know about additional regional parameter dependencies, disabling non-regional parameter optimisation should reduce the complexity of the search space, resulting in ‘independently’ tunable regional parameters.

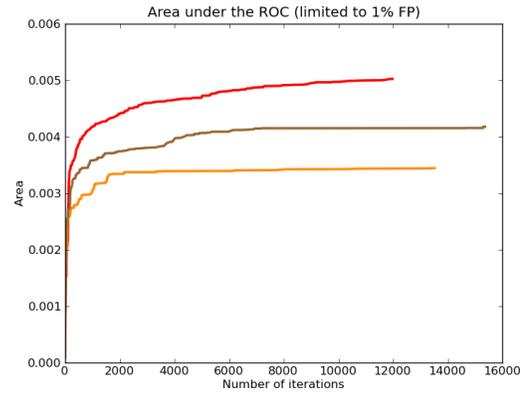
6.4.2 Single-parent, multi-archive, non-regionals non-optimisable

Here we take the poorly-performing single-parent, multi-archive optimisation scheme and see whether the current best (multi-parent, multi-archive; section 4.1.2, page 65) can be equalled by disabling optimisation of the non-regional parameters. Throughout the optimisation the non-regional parameter values are fixed to those defined in the NATS operating point. Figure 6.7(a) compares the result of an optimisation. The light brown squares illustrate the state of the ROC after 11900 iterations on two weeks corrected MACC data, single-parent, multi-archive, non-regionals disabled. The optimiser has located 30 parametrisations ranging from 6.53% to 51.26% True Positive and 0.01% to 0.93% False Positive. The trial was initialised with a single randomised parametrisation. The orange squares are the single-parent, multi-archive run from section 6.3.1 (non-regionals enabled for optimisation). Our ‘current best’ front from the multi-parent, multi-archive run (section 4.1.2) is shown as red squares.

As expected, disabling optimisation of the non-regional parameters has resulted in a significant improvement in the efficacy of the single-parent optimiser; the single-parent optimiser has even managed to locate some parametrisations that dominate those on the multi-parent Pareto Front (between $\approx 14\%$ and $\approx 35\%$ TP). However, from $\approx 35\%$ TP upwards the single-parent ROC still rapidly tails off, showing signs of stunted growth and is ultimately significantly dominated by the multi-parent (red squares) ROC. Referring to figure 6.7(b) which shows the area under the ROC, we can see that progression does appear to have stalled (the brown line) and that the ROC has converged as far as possible given the current settings. The NATS operating point has still not been located.



(a) ROC.



(b) Area under ROC (limited to 1% FP).

Figure 6.7: Comparison of single-parent (orange), single-parent non-regional parameters non-optimisable (brown), and multi-parent (red) multi-archive optimisation results, evaluated on the MACC dataset with corrected regions (11900 iterations). The (randomised) starting points for single-parent runs have been labelled ‘SP’.

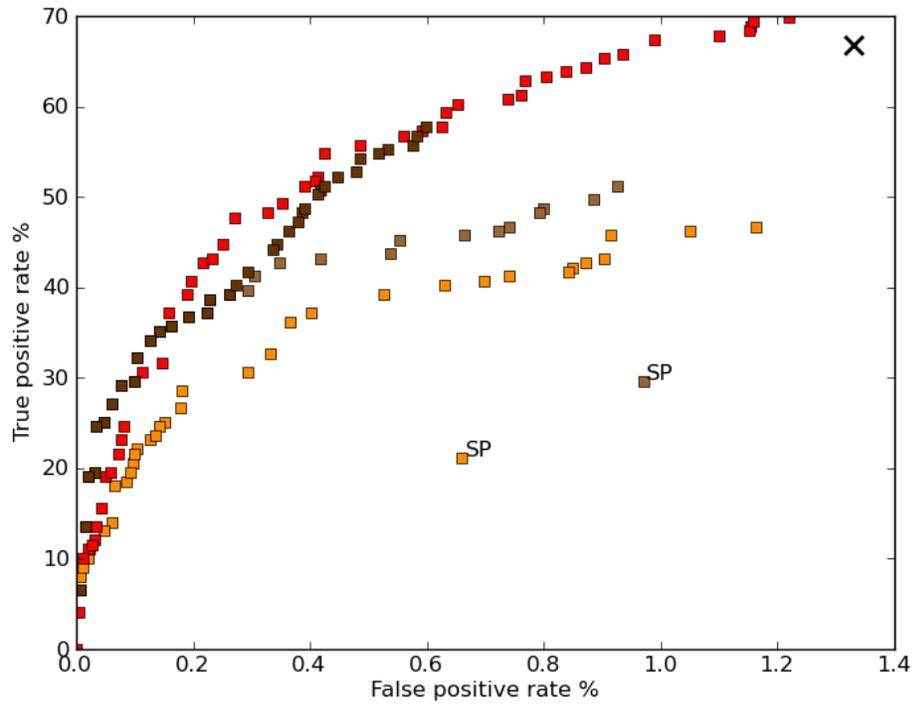


Figure 6.8: Comparison of convergence after the non-regional parameters were re-enabled for a further 5000 iterations (dark brown) following initial optimisation which achieved the Pareto front shown in light brown.

In figure 6.8 we show the result of continuing the optimisation of the ‘single-parent, multi-archive, non-regionals disabled’ trial, this time with optimisation of the non-regional parameters *enabled* for a further 5000 iterations (16900 iterations in total). We did this to allow the optimiser to make some final ‘tweaks’ to the non-regional parameters. This additional freedom has allowed the stalled ROC to progress much further (dark brown squares), equalling the multi-parent ROC in several places. The coverage of the single-parent ROC is still, however, not as extensive as that of the multi-parent and the NATS operating point has *not* been located or exceeded.

Given that disabling/enabling non-regional parameter optimisation is highly beneficial to the single-parent optimiser (despite it still not bettering the multi-parent scheme), we next apply the procedure to the multi-parent, multi-archive optimiser.

6.4.3 Multi-parent, multi-archive, non-regionals non-optimisable

Figure 6.9 depicts two multi-parent, multi-archive optimisation trials. It compares the result of disabling optimisation of non-regional parameters (dark red) against a previous run with them enabled (lighter red, see section 4.1.2, page 65). Both trials were initialised with 100 randomised parametrisations on two weeks corrected MACC data and allowed to run for 11900 iterations. With optimisation of non-regional parameters disabled, the optimiser has located 42 parametrisations ranging from 12.56% to 68.84% True Positive and 0.01% to 0.94% False Positive. It appears that the new parametrisations (dark red) almost exclusively dominate those found by optimisation with non-regional parameters enabled (lighter red); the ROC generated by optimisation with non-regional parameter perturbation disabled lies significantly ahead of that generated with non-regional parameter perturbation enabled. The NATS operating point has been dominated; for a similar TP rate of $\approx 67\%$ the optimiser has located a parameterisation with a FP rate of 0.71% (0.62% less than the NATS COP). This constitutes a new ‘best case’.

In figure 6.10 we continue optimisation of the ‘multi-parent, multi-archive, non-regionals disabled’ trial, this time with optimisation of the non-regional parameters *enabled* for a further 5000 iterations (16900 iterations in total). Unlike the single-parent case (section 6.4.2), the additional convergence attained (yellow squares) is not as startling. It is possible that this estimated Pareto front is now very close to the *true* Pareto front and thus larger jumps in convergence are not feasible.

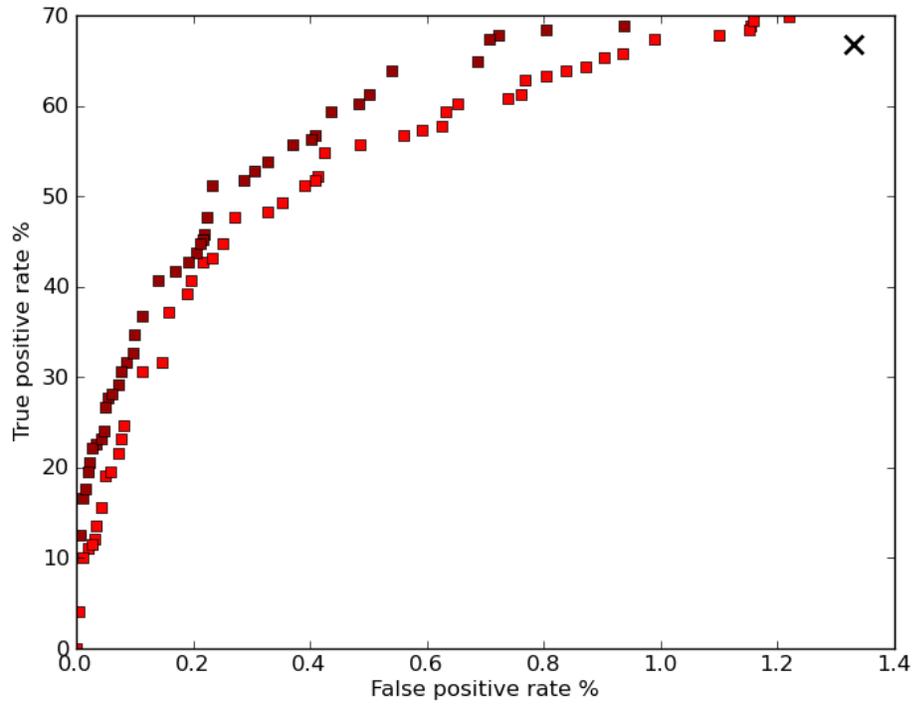


Figure 6.9: Comparison of multi-parent (red), and multi-parent non-regional parameters non-optimisable (dark red) multi-archive optimisation results, evaluated on the MACC dataset with corrected regions (11900 iterations).

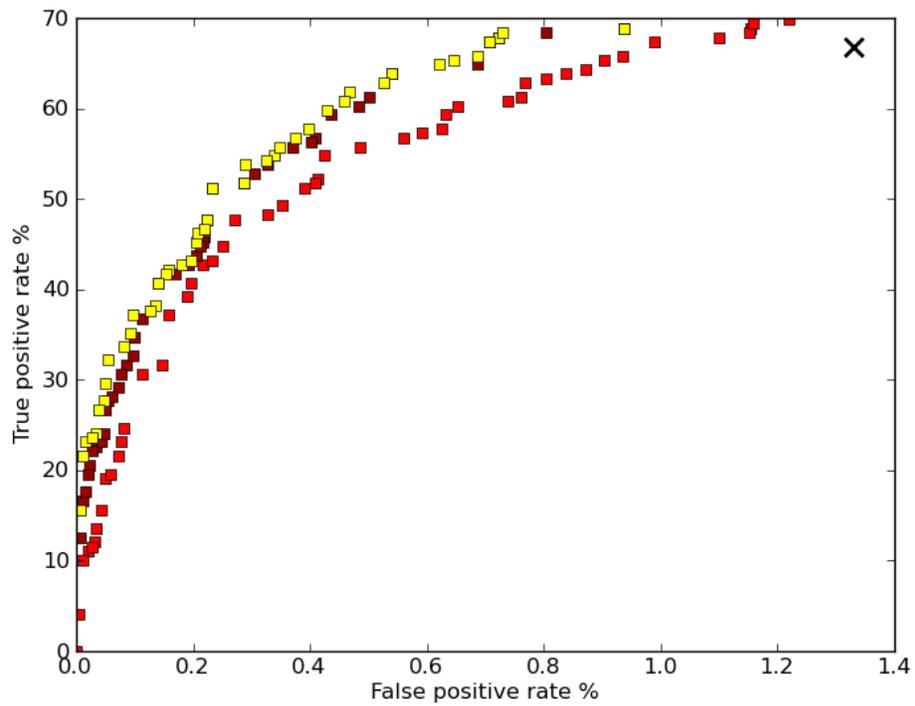


Figure 6.10: Comparison of convergence after the non-regional parameters were re-enabled for a further 5000 iterations (yellow) following initial optimisation which achieved the Pareto front shown in dark red.

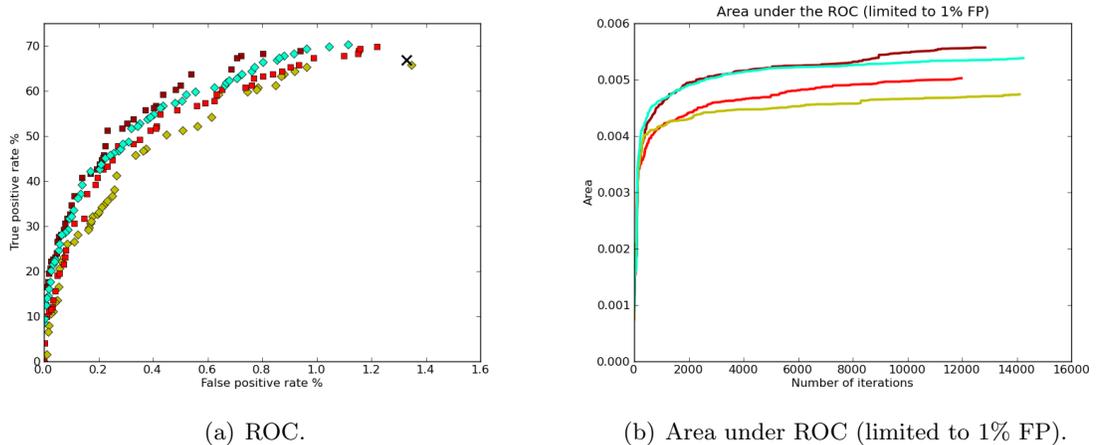


Figure 6.11: Comparison of multi-parent multi-archive (red), multi-parent multi-archive with non-regional parameters non-optimisable (dark red), the original estimated-archive (yellow) and estimated-archive run with non-regional parameters non-optimisable (light blue), MACC dataset (11900 iterations) with corrected regions.

6.4.4 Estimated-archive, non-regionals non-optimisable

For the sake of completeness the original estimated-archive optimiser (section 4.2, page 68) was also re-run with optimisation of non-regional parameters disabled. Figure 6.11(a) compares this trial (light blue diamonds) to the original estimated-archive results (yellow diamonds; section 4.2.1, page 72), the previous best multi-archive case (red squares; section 4.1.2, page 65) and the new best case; multi-archive with non-regional parameters non-optimisable (dark red squares; section 6.4.3). All trials were initialised with 100 randomised parametrisations on two weeks corrected MACC data and allowed to run for 11900 iterations.

With optimisation of non-regional parameters disabled, the estimated-archive scheme has located an actual (non-estimated) front consisting of 60 parametrisations ranging from 8.54% to 70.35% True Positive and 0.00% to 1.12% False Positive. For the first time the estimated-archive has matched (and in the majority of places actually dominated) our previous best (multi-archive mode; red squares). In addition, the ROC generated is much less sparse, with 60 parametrisations compared to 42 in the multi-archive case. This corroborates our speculation that the estimated-archive suffers from dependencies related to the non-regional parameters (see section 4.3.3, page 81). Figure 6.11(b) confirms that, with non-regional parameter optimisation disabled, the estimated-archive (light blue line) is capable of bettering the convergence rate of the multi-archive optimiser (red line). However, when the multi-archive optimiser is also run with non-regional parameters disabled (dark red line), the estimated-archive is once again shown to be bettered. Interestingly, up to ≈ 6500 iterations the estimated-archive is actually roughly equal to multi-archive mode in terms of convergence (and even betters it in the early stages between ≈ 0 and ≈ 1500 iterations). However, at ≈ 6500 iterations the estimated-archive's convergence rate levels more rapidly than that of the multi-archive.

6.4.5 Analysis of non-regional parameter values

The results presented in the previous sections highlight the significance of the non-regional parameters in optimisation efficacy. Since these (comparatively few) parameters hold such importance, we investigate their configuration in this section. Of the ≈ 36 non-regional parameters, only 6 have been marked for optimisation throughout our trials, further emphasizing their impact. Most of the remaining non-regional parameters are either physical constraints or statutory requirements and thus should not be perturbed.

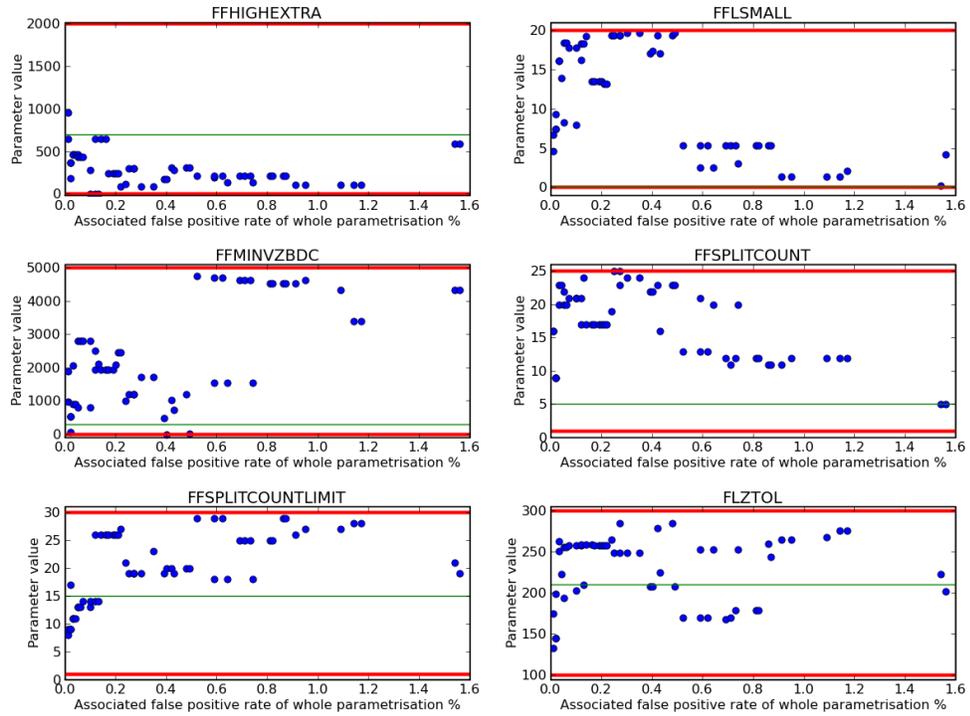
The 6 non-regional parameters marked for optimisation are as follows:

- FFHIGHEXTRA – adds extra height to the threshold for vertical separation tests. Only applicable to high-level encounters (of which there are relatively few).
- FFLSMALL – defines the minimum separation for the calculation of aircraft closing speed. Affects the number of aircraft pairs that get passed on to the ‘current proximity filter’.
- FFMINVZBDC – modifies the behaviour of a flag that defines whether aircraft are climbing or descending. Mainly affects the ‘linear prediction filter’.
- FFSPLITCOUNTLIMIT – determines the limit at which aircraft tracks are considered invalid due to bad radar returns.
- FFSPLITCOUNT – modifies the behaviour of the procedure that keeps a history of aircraft track validity. If tracks are in a certain range the method will add to the ‘split count’, else it will subtract, up until FFSPLITCOUNTLIMIT is reached.
- FLZTOL – determines the tolerance given to a flight level limit. Permits ‘buffering’ of fluctuations in an aircraft’s track so that it is not erroneously categorised as climbing or descending.

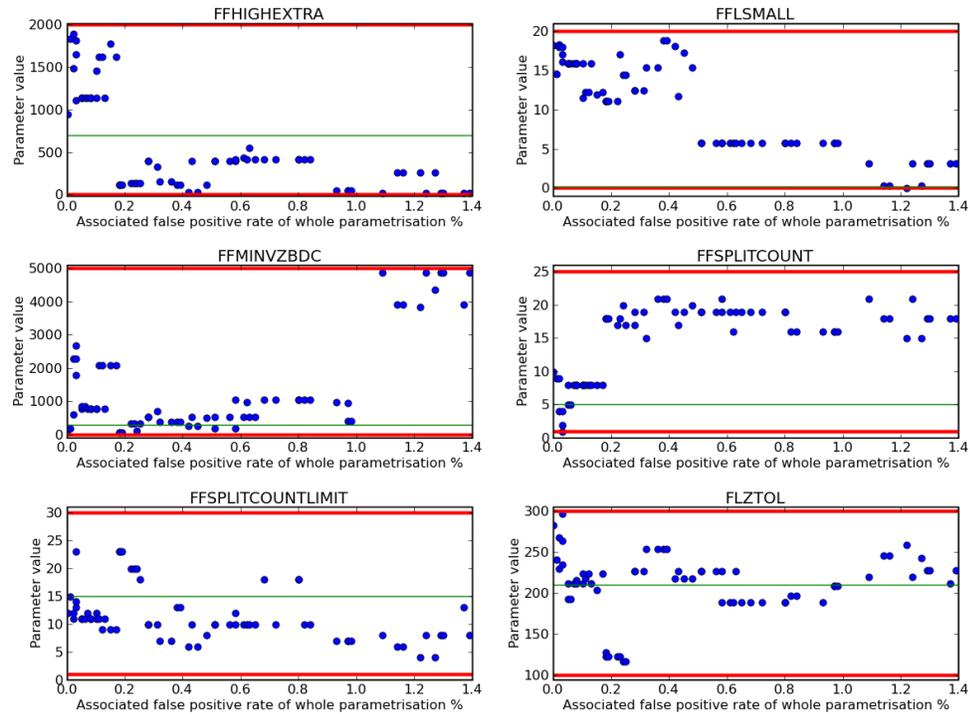
Figure 6.12 compares non-regional parameter value ranges taken from parametrisations found on the optimised Pareto Fronts of the single-archive optimiser in single (figure 6.12(a)) and multi (figure 6.12(b)) parent modes. Parameter values are plotted versus the corresponding parametrisation’s overall False Positive rate (%). The ‘conservative’ minimum and maximum boundaries for each parametrisation are marked as a horizontal red line. The green horizontal line marks the parameter value as defined in the NATS operating point. Figure 6.13 compares similar results obtained from parametrisations found on the optimised Pareto Fronts of the *multi*-archive optimiser (again, single (figure 6.13(a)) and multi (figure 6.13(b)) parent modes are illustrated). For the purposes of comparison ‘optimised fronts’ are those as achieved after 11900 iterations in each case.

It is hard to discern any firm trends from these plots; however, there is a slightly larger degree of scatter in the multi-archive cases (particularly multi-parent, multi-archive, figure 6.13(b)), indicating a greater diversity in the population of solutions. Interestingly, there are areas in which the parameter values are fairly consistent, for example figure 6.12(a);

FFHIGHEXTRA a value of ≈ 250 , figure 6.12(b); FFSPLITCOUNT a value of ≈ 19 , figure 6.12(b); FFLSMALL a value of ≈ 5 , figure 6.13(b); FFLSMALL also a value of ≈ 5 , despite large changes in the associated percentage False Positive rate, perhaps indicating the location of a near optimal value in the context of the population of parametrisations as a whole. However, the location of these ‘optimum’ values is largely inconsistent between optimisation modes and in each case likely originates from an early parent parametrisation configuration ‘living on’ as subsequent child generations are ‘tweaked’. This perhaps indicates the presence of unknown parameter interactions as there may exist more than one solution for achieving a given TP/FP rate. Indeed, the plots illustrate that there are parametrisations for which the FP rate is almost identical but associated non-regional parameters are observed to vary extensively in value; for example, see figure 6.13(a); FLZTOL ≈ 1.5 FP, figure 6.13(b); FFSPLITCOUNTLIMIT ≈ 3.5 FP. Therefore, there must either be some compensating parameters elsewhere that keep the FP rate constant, or the non-regional parameter concerned is not contributing to the false positive rate in those instances.

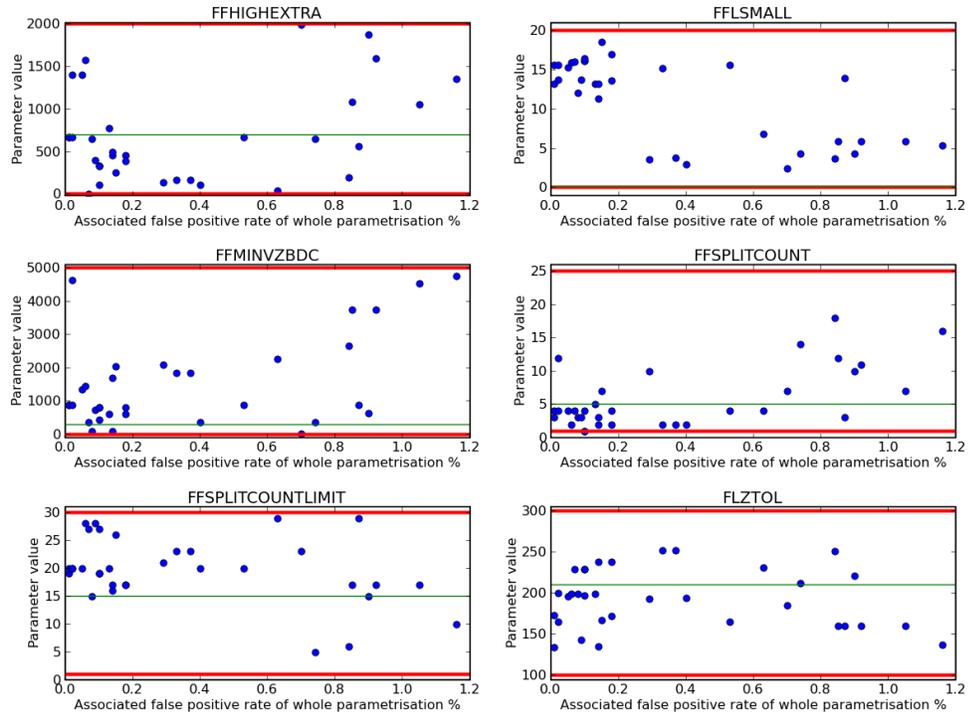


(a) Single-parent, single-archive.

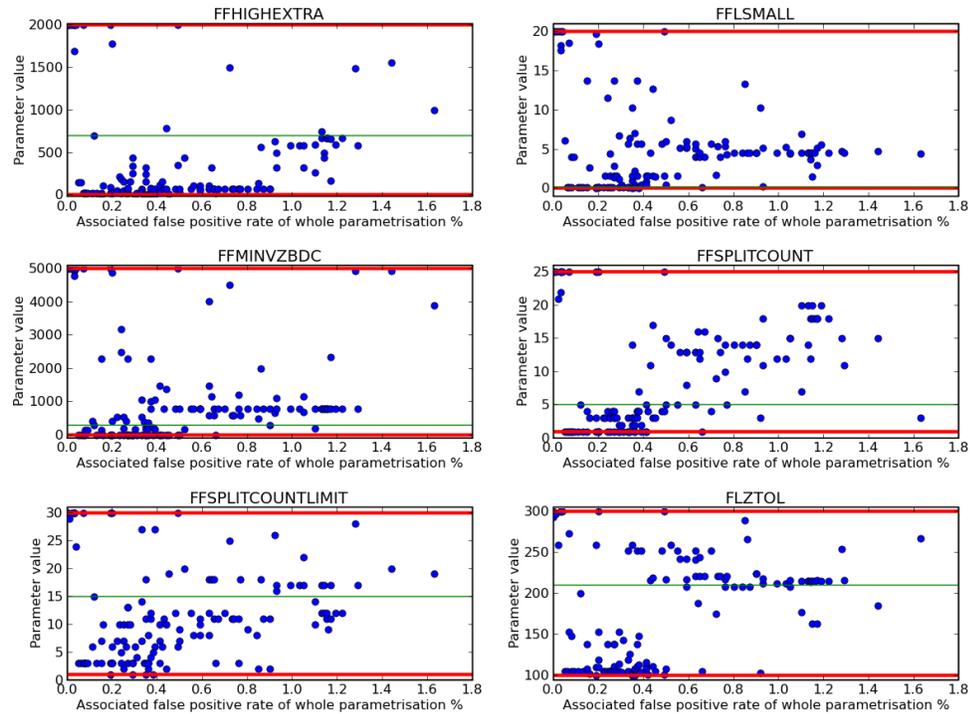


(b) Multi-parent, single-archive.

Figure 6.12: Plots of the non-regional parameter values for solutions along the Pareto front, ordered by the corresponding parametrisation's false positive rate. Each panel corresponds to the indicated parameter. Green lines show the NATS current operating point. Red lines show the conservative min/max ranges.



(a) Single-parent, multi-archive.



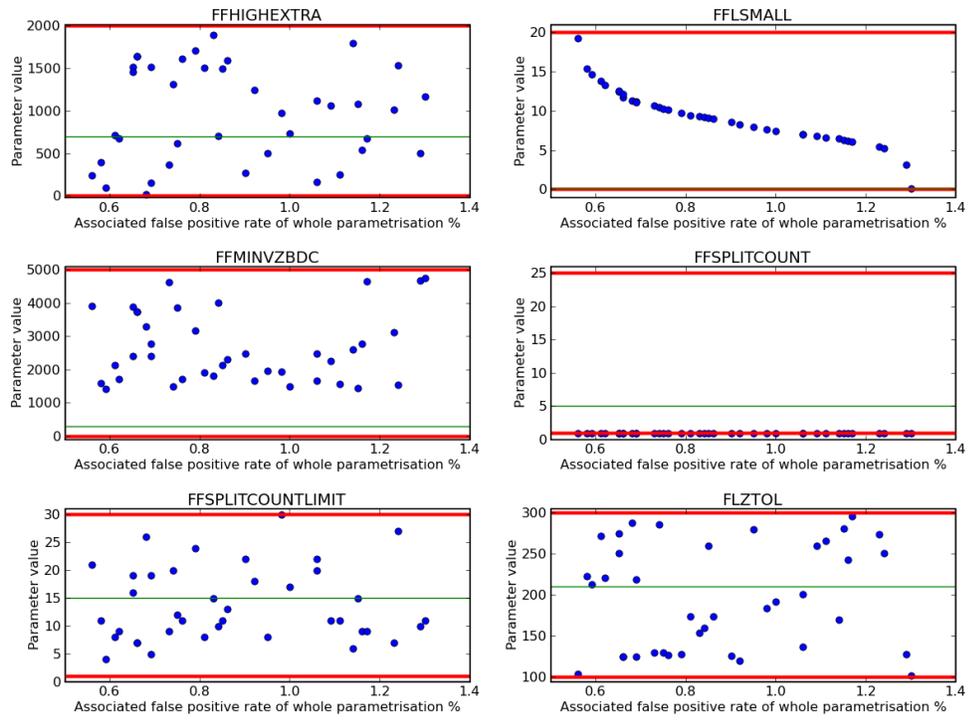
(b) Multi-parent, multi-archive.

Figure 6.13: Plots of the non-regional parameter values for solutions along the Pareto front, ordered by the corresponding parametrisation’s false positive rate. Each panel corresponds to the indicated parameter. Green lines show the NATS current operating point. Red lines show the conservative min/max ranges.

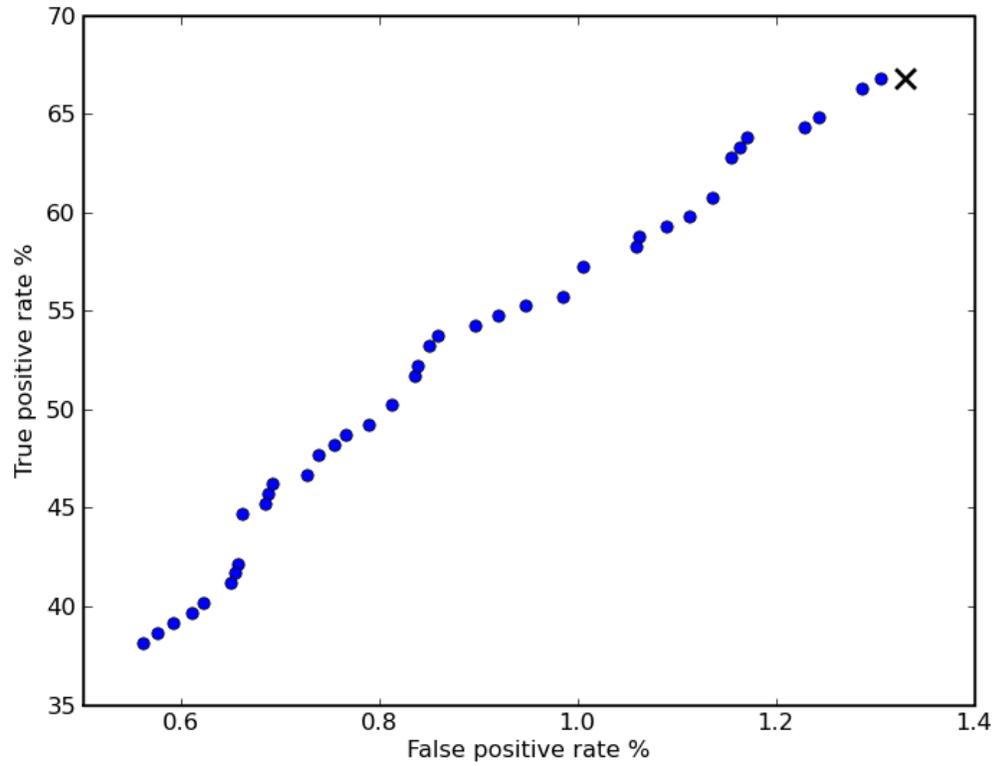
In the hope of further assessing non-regional parameter contributions a trial was run where *only* the 6 non-regional parameters were optimised; all other parameters were fixed to the NATS operating point values. The range of parameter values and corresponding FP rates generated after 11900 iterations may be observed in figure 6.14(a). The corresponding ROC from which these points were plotted is shown in figure 6.14(b). Here we see a very clear trend between FP rate and the parameter FFLSMALL. High values of FFLSMALL lead to lower FP rates whilst low values of FFLSMALL result in higher FP rates.

On examining the role of FFLSMALL in STCA it becomes clear why this parameter is so significant; FFLSMALL defines the minimum separation for the calculation of aircraft closing speed. Essentially it directly affects the number of aircraft pairs that get passed on to the ‘current proximity filter’. If the min-separation test fails (does not trigger), then the pairs concerned will not be further processed by the current proximity filter. This means that by modifying the value FFLSMALL we can essentially include or exclude a large subset of aircraft pairs from further examination. Indeed all 6 non-regional parameters (excluding FLZTOL) are concerned with STCA’s fine filter preprocessing stage, thus explaining why these comparatively few parameters can have such effect on TP/FP rates.

As figure 6.14(a) indicates that FFLSMALL is the non-regional parameter of greatest significance, dominating the effect of all others, a trial was run to see what range of TP/FP rates could be generated by modifying this parameter alone. Results of varying FFLSMALL between 0 and 20 can be seen in figure 6.15. It is interesting to note the fairly diverse range of TP/FP rates that can be obtained by changing this single parameter. However, no amount of adjustment of FFLSMALL, with other parameters fixed at the COP values, will yield higher TPR than that of the COP itself.

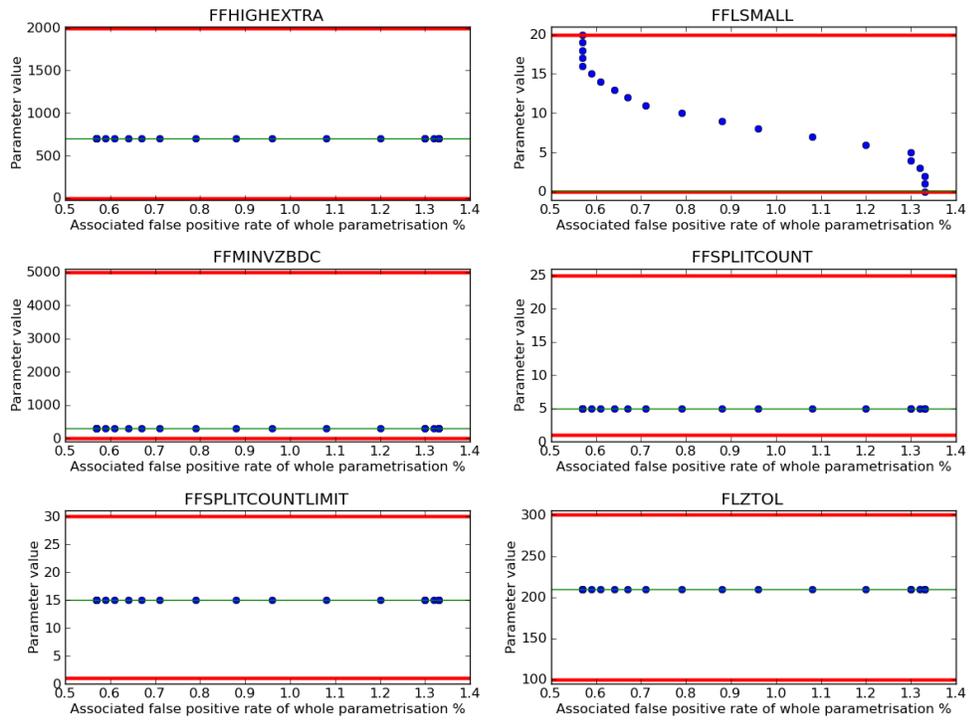


(a) Non-regional parameter value ranges by corresponding parametrisation's FP rate.

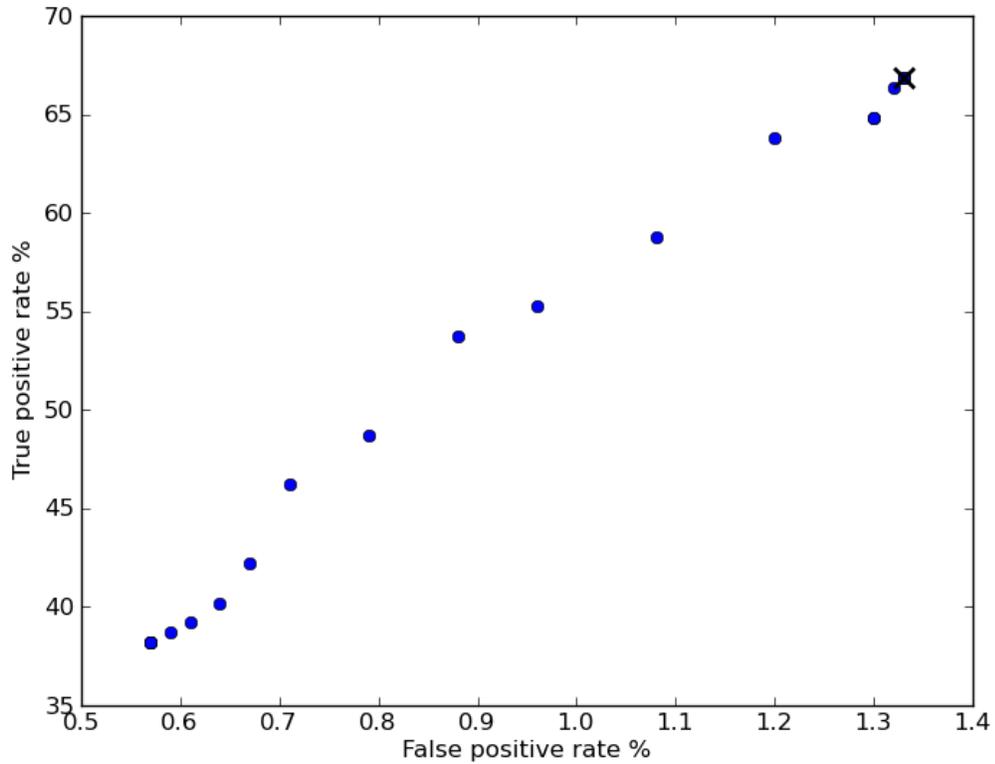


(b) ROC.

Figure 6.14: Result of optimising *only* the non-regional parameters (all other parameters are set to NATS operating point values).



(a) Non-regional parameter value ranges by corresponding parametrisation's FP rate.



(b) ROC.

Figure 6.15: The effect of sweeping the value of FFLSMALL between 0 and 20 (all other parameters are set to NATS operating point values). Note the range of TP/FP rates that can be obtained by changing this single parameter.

6.4.6 Summary

In the preceding section we have investigated the effect of non-regional parameter dependencies on multi-archive (single, and multi-parent) schemes. Results have indicated that disabling optimisation of non-regional parameters can benefit the rate of multi-archive optimiser convergence in the single-parent and multi-parent cases. Subsequently enabling optimisation of non-regionals for a further ≈ 5000 ‘fine tuning’ iterations is beneficial in the single-parent case and does not adversely affect the multi-parent optimiser. However, as the ROC appears to have converged comparatively early on, re-enabling optimisation of non-regionals provides no major advantage in the multi-parent case. Similarly, disabling optimisation of non-regional parameters has shown a positive effect on the estimated-archive, generating an ROC roughly equivalent to that of the multi-parent, multi-archive optimiser with non-regional parameter optimisation enabled.

Removing the dependency of non-regional parameters appears to benefit both single and multi-parent schemes in early iterations. We could therefore recommend that an optimisation procedure of ≈ 12000 iterations with non-regional parameter optimisation disabled, followed by a ‘fine tuning’ phase of ≈ 5000 iterations (non-regional parameter optimisation enabled), is most effective in speeding up ROC convergence when initialising with randomised parametrisations. These results do not, however, conclusively indicate that restriction of regional parameter dependencies plays the biggest role in speed of convergence. One could argue that it is the reduction in the number of parameters (and consequently the reduction in search space complexity) that leads to faster convergence. Even though, in actuality, there were only 6 non-regional parameters (out of a total ≈ 1000) enabled for optimisation throughout our trials, their significance and ability to have a vast effect on TP/FP rates has been shown in section 6.4.5.

One could also argue that the benefit in ROC convergence, seen when optimisation of non-regionals is disabled, may be attributed to the fact that these (not insignificant) parameters will have starting values appreciably better than those of the initially randomised parameters in other non-regional enabled trials. That is, by fixing the non-regional parameters to the NATS OP values (when non-regionals are disabled from optimisation), the optimiser will have a better starting point, and thus converge quicker. Therefore, some of the expected benefits we might like to attribute to a reduction in regional parameter dependencies, may in fact be more realistically attributed to use of the NATS Operating Point values.

Thus, when tuning already established airspaces fixing non-regional values, at least initially, should be of benefit to convergence. However, when learning parameter values in newly configured airspaces, for which there is no previous optimal/near optimal precedence, optimisation of non-regional parameters may have to be permitted from the start, and so we would expect a slower convergence rate.

6.5 Chapter summary

At the beginning of this chapter we began by investigating an alternative parental selection scheme. Research by [72] and [61] seemed to suggest that a ‘single-parent’, kept and perturbed throughout all iterations, may be beneficial to the rate of ROC convergence. When applied to the task of optimising STCA single/multi-parent, single-archive optimisation was found to be roughly equivalent. In the multi-archive case, use of a single-parent was found to be severely detrimental; possibly due to a reduction in population diversity. Both single and multi-archive, single-parent optimisations were comparatively erratic in ROC progression compared to their multi-parent counterparts. We speculate that this was caused by single-parent mode getting ‘stuck’ at a particular parametrisation. The multi-parent, multi-archive scheme is still found to be superior over all other modes examined thus far.

In section 6.4 we looked again at a potential source of dependency between regional parameter subsets; the non-regional parameters. By disabling optimisation of these, comparatively few, parameters we hoped to reduce one source of noise. Results showed that all optimisation modes examined did indeed benefit from disabling optimisation of the non-regional parameters, however, we could not conclusively determine whether the increase in ROC convergence rates seen were due to a reduction in noise caused by regional parameter dependencies, a reduction in search space complexity, or use of NATS OP values substituted into the non-regional values.

In the next chapter we investigate how well optimised parametrisations generalise to unseen data. We examine how variance between training and test TP/FP rates might be reduced and discuss how such methods might be applied in optimisation of STCA.

Chapter 7

Classifier over-fitting

Overfitting, whereby an ‘optimum’ parametrisation becomes too adapted to training data and does not generalise well to ‘unseen’ instances, is a potential problem for all machine learning techniques. In this chapter we examine how much the performance of a classifier that is prone to overfitting varies when exposed to unseen data. We discuss the impact of such effects on the real-world STCA example, before experimenting on a ‘toy’ problem, which is orders of magnitude faster to evaluate and with known decision solutions, in order to devise and assess techniques designed to alleviate the issue.

7.1 STCA classifier variance

Figure 7.1 shows a comparison plot of the multi-archive, aggressive, estimated Pareto Front after re-evaluation on two weeks of ‘unseen’ data. The original TP/FP rates as observed during optimisation are shown in light green, re-evaluated rates are plotted in dark green. The positions of the re-evaluated solutions in objective space are far outside the expected variance; two standard deviations estimated using bootstrapping of results [35, 26] and plotted as red error bars. They are thus statistically significant, implying that the training dataset is not wholly representative and also that the classifier (STCA) has been over-trained during optimisation; the solutions are overfitting. It is clearly the case that the two datasets are different, if they were statistically the same then the re-evaluations would lie inside the two standard deviation error bars. However, over-training is also indicated by the fact that performance on the second set is *worse*, especially when considering that the NATS manually tuned operating point actually increases in efficacy on the second two weeks data. It is interesting to note that there is comparatively little variance in the FP rate, as is perhaps expected given the vastly skewed ratio of wanted to unwanted alert examples on which to train.

Perhaps the easiest solution to overfitting is to provide a larger training dataset that is better representative of broader trends to be modelled, however this is not always possible. Indeed, one issue for optimisation of STCA is that the datasets used already contain a large number of instances that must be manually confirmed. Increasing the size of the dataset

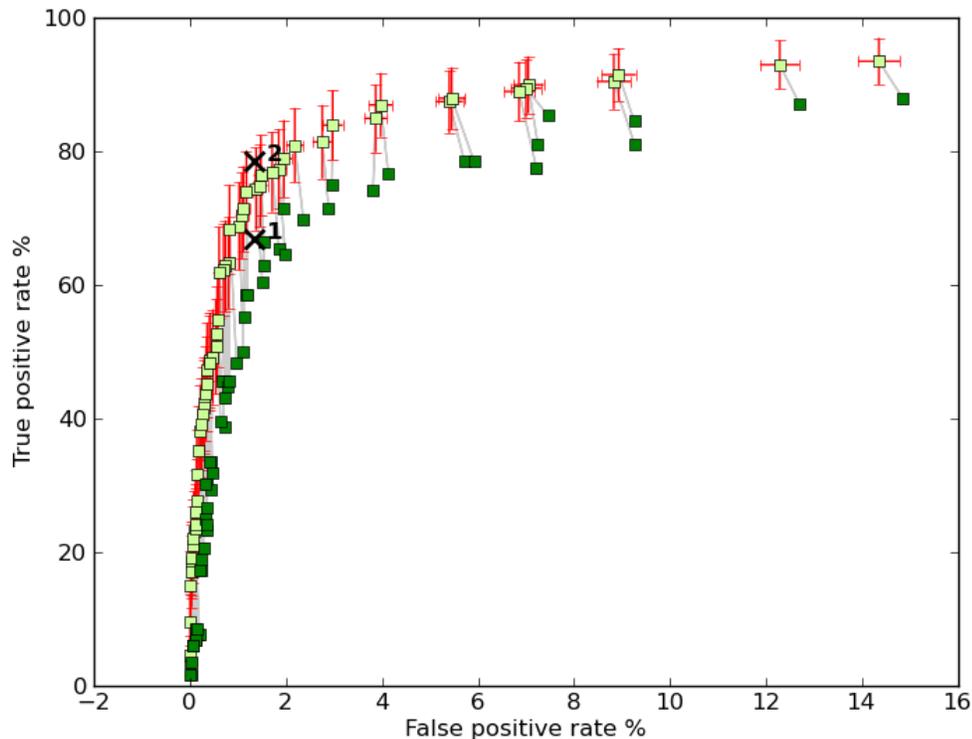


Figure 7.1: Comparison of multi-archive aggressive optimisation results, MACC dataset (14402 iterations), as optimised on weeks 1 and 2 dataset (light green, corresponding OP marked as X1) and re-evaluated on weeks 3 and 4 dataset (dark green, corresponding OP marked as X2). Two standard deviations have been marked as red error bars.

amounts to a heavy work load increase for personnel analysing each potential ‘true positive’ aircraft encounter. The purpose of automating optimisation of STCA is partly to achieve an increase in system performance but is also very much aimed at cutting the laborious manual optimisation process. A requirement to obtain more data clearly undermines this. Thus, in the next section we will look at some potential methods to combat overfitting, without increasing training dataset size.

7.2 Experimentation on a ‘toy’ problem

When running multiple experimental trials it is preferable that a ‘toy’ problem be used. Using a simplified classifier is computationally cheaper and allows us to run many tests quickly without requiring specialist hardware, unlike the STCA/ARDAT setup which can take several weeks to generate a single result. We implement a neural network and train it using an EA to classify unseen instances of data generated by a Gaussian mixture model. Traditionally supervised learning techniques such as the ‘backpropagation algorithm’ [10, 25], that use a gradient descent approach to minimising the network error, have been used to effectively optimise neural networks. However, as we are seeking solutions to overfitting EAs and to

mimic the optimisation of TP and FP rather than only accuracy, we substitute a basic (1 + 1) EA to perform this task.

The dataset

The classification task is to correctly predict the class label of a data-point based on a vector of input features (location parameters), \mathbf{x} . A set of five Gaussian distributions, forming a Gaussian mixture model, are sampled to provide data-point locations; one of two class labels is assigned to each distribution. To make the problem more challenging the distributions are overlapping generating a complex boundary between classes for which a ‘perfect’ solution that correctly segregates all members does not exist. This is a variation of the Ripley toy dataset [70]; an additional Gaussian distribution has been added and the location parameters adjusted to increase overlap. In later trials we also expanded the features from two dimensions into four. Because we know the model’s parameters, the Bayes’ error rate (which is the best error rate that can be achieved) can be calculated. We can thus compare outputs of our trained neural network with a known ‘true’ optimal ROC.

The test problem comprises two classes, each of which is generated by a Gaussian mixture model. The density of each class is given by:

$$p(\mathbf{x}|C_j) = \sum_i w_{ji} \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_{ji}, \sigma^2 \mathbf{I}) \quad (7.1)$$

where $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the Gaussian density with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. The parameters used were $\sigma^2 = 0.03$ and for class 1:

$$w_{11} = 0.32 \quad \boldsymbol{\mu}_{11} = (0.5, 0.5, 0.1, 0.0) \quad (7.2)$$

$$w_{12} = 0.34 \quad \boldsymbol{\mu}_{12} = (-0.7, 0.3, 0.0, 0.0) \quad (7.3)$$

$$w_{13} = 0.34 \quad \boldsymbol{\mu}_{13} = (0.3, 0.3, 0.1, 0.0) \quad (7.4)$$

For class 2:

$$w_{21} = 0.7 \quad \boldsymbol{\mu}_{21} = (-0.5, 0.5, 0.0, 0.0) \quad (7.5)$$

$$w_{22} = 0.3 \quad \boldsymbol{\mu}_{22} = (-0.2, 0.4, 0.0, 0.0) \quad (7.6)$$

Figure 7.2 shows an example two dimensional configuration of the dataset. This particular composition was used in some early experiments but was found not to sufficiently over-fit, thus the 4D dataset was ultimately used. Illustrating a configuration of this dataset in four dimensions becomes problematic, so we have included this 2D counterpart simply as an example. In this instance, to approximate the class skew observed in STCA data the ratio of ‘wanted’ instances (white circles) to ‘unwanted’ (green squares) has been set to 1 : 100. However, unlike STCA, when using this simplified classification scenario, it was found that the negative class instances actually provide a lot of information about the boundary with the positive class, thus in actuality the effect of class skew is not as representative of the

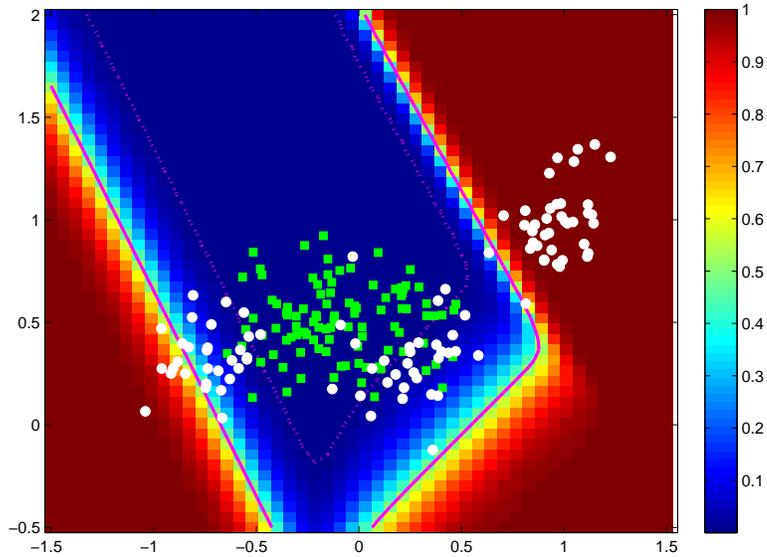


Figure 7.2: Example ‘toy’ dataset. Wanted instances are marked as white circles, unwanted as green squares. The ratio of wanted to unwanted instances in the training dataset is 1 : 100, however for clarity here we have plotted them at a ratio of 1 : 1.

STCA problem as we might like. The pink lines in figure 7.2 show the Bayes optimal decision boundary with equal costs of misclassification (solid line) and the boundary if the ratio of the misclassification costs is 1000 : 1 (dashed line) – i.e, if it were 1000 times more costly to misclassify the white circles class than the green squares class.

The neural network

The concept of neural networks is inspired by the biological interconnection of neurons observed in nature e.g. in the brain. A series of interconnected nodes is used to model relationships between inputs and outputs. The neural network we implement is a feed-forward (no cycling between nodes) multi-layer perceptron as illustrated in figure 7.3. Inputs are the set of location parameters \mathbf{x} (2D or 4D), from which a single output value is ultimately generated (in the range 0–1). The output is then thresholded at 0.5 to generate a binary classification. We use a flexible single layer network capable of modelling any function to arbitrary accuracy [20].

Our network includes a single hidden layer of perceptrons; each of these is a hyperbolic tangent sigmoidal transfer function. Increasing the number of hidden nodes can increase the flexibility of a network; ≈ 4 nodes would usually be adequate for our classification problem, however, to ensure over-fitting we use 20. Connections between each node in the network are weighted; simulating the function of synapses altering signal flow in the brain. It is these weights that must be adjusted by the EA in order to train the network.

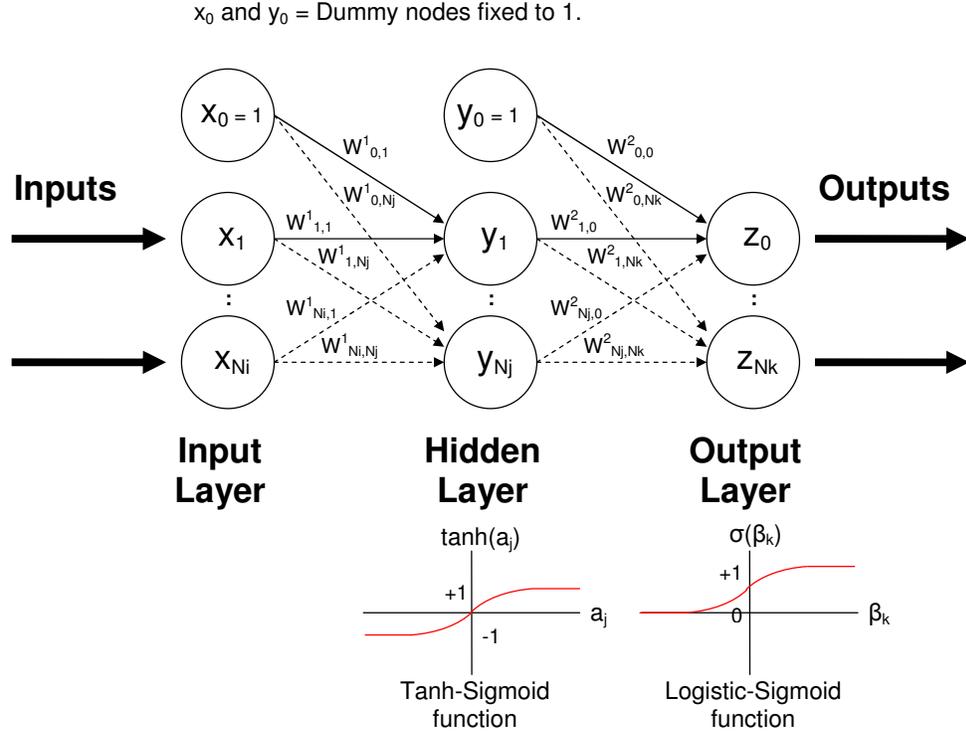


Figure 7.3: Illustration of the architecture of a multi-layer perceptron.

Referring to figure 7.3, the ‘activation’ value, a_j , passed to hidden node, y_j , is calculated as:

$$a_j = \sum_{i=0}^{N_i} w^1_{ij} x_i \quad (7.7)$$

where w^1_{ij} are the input layer weights. The sigmoid transfer function at hidden node, y_j , is calculated as:

$$y_j = \tanh(a_j) \quad (7.8)$$

The input, β_k , to the output node, z_k , is calculated as:

$$\beta_k = \sum_{j=0}^{N_j} w^2_{jk} y_j \quad (7.9)$$

and the logistic output of node, z_k , is calculated as:

$$z_k = \sigma(\beta_k) \equiv \frac{1}{1 + \exp(-\beta_k)} \quad (7.10)$$

For extensive information on neural networks see, for example, [70] or [10].

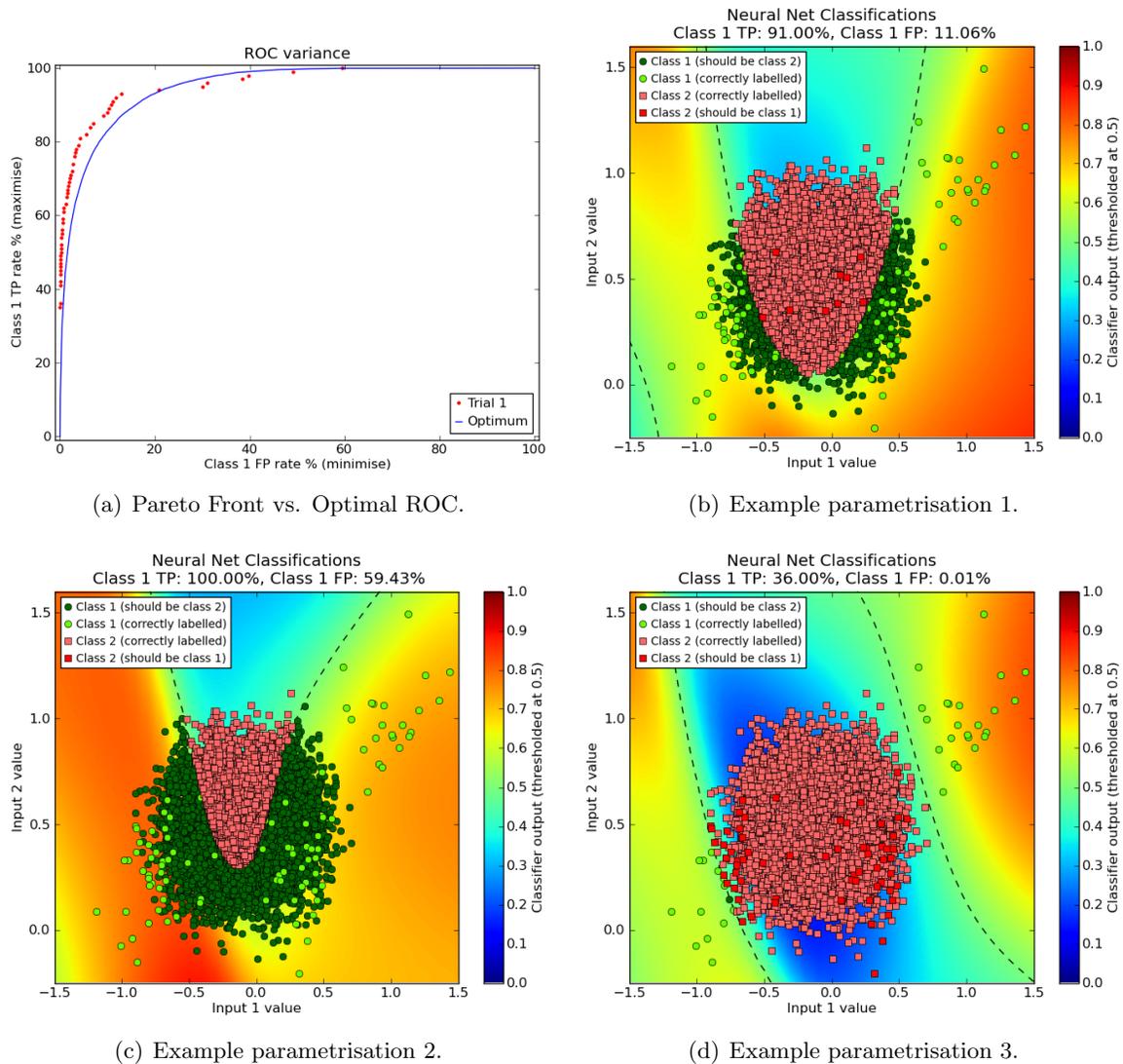


Figure 7.4: Example output from a neural network trained on a toy dataset using an EA. In 7.4(a) the solid blue line indicates the Bayes optimal ROC curve. In figures 7.4(b), 7.4(c) and 7.4(d), a trained classifier’s posterior probability of the positive class is shown by the colour scale and the dashed black line shows the estimated decision boundary. These figures illustrate classifiers selected from three points along the trained ROC curve shown in 7.4(a). The colour of each of the plotted symbols indicates whether a data point was correctly classified.

Some example parametrisations taken from a trained (2D input) network are illustrated in figure 7.4. Figure 7.4(a) shows the Pareto front from which these parametrisations were selected; the EA’s archive (marked as ‘Trial 1’) has not over-fitted as much as desired for the purposes of these trials. All plots are generated from training data containing 100 class 1s (designated as the ‘wanted’ or positive class) and 10,000 class 2s (the ‘unwanted’ or negative class). This particular neural network contained 10 hidden nodes and was optimised over 10,000 iterations. In each case the classifier’s decision boundary has been marked as a dashed black line.

Increasing over-fitting

In the following sections we present seven approaches to optimising a neural network using an evolutionary algorithm, assessing each for its efficacy in reducing over-fitting and variance when parametrisations are evaluated on unseen data. To increase over-training we run the trials for 50,000 EA iterations and allow 20 hidden layer nodes for extra flexibility in the neural network. The training dataset has also been reduced, containing 100 class 1s (designated as wanted) and 900 class 2s (designated unwanted) – a ratio of 1 : 9. The ratio of wanted to unwanted classes is still skewed but has been drastically reduced as in initial trials it was found that, on this simplified problem, the negative class instances can contribute a lot of information about the boundary with the positive class, unnecessarily increasing the dataset size without providing the expected class-skew modeling we might like.

Each trial is initialised with 100 randomly generated parametrisations (neural network weights). The probability of selection is set to 0.2 (20% of the weights will be perturbed each iteration). Perturbation itself is based on a Laplacian distribution with width 0.1.

To summarise, we encourage over-training of our neural network in the following ways:

- Using a smaller training dataset: reducing the size of a dataset may mean that it becomes less representative of broader trends, increasing the chances of noise adversely affecting a classifier’s development. We use a dataset with a total of 1000 members. This is actually still quite a large dataset but it is necessary because of the class skew.
- Using a large number of hidden layer nodes: a large number of nodes results in a classifier function that is very flexible, the ability to generate many boundary curves decreases generalisation and generates a network that can over-fit around individual data points. We allow 20 hidden layer nodes.
- Heeding ‘the curse of dimensionality’ [10]: the term originally coined by Richard Bellman refers to the exponential increase in volume associated with adding extra dimensions to a search space. In higher dimensions it becomes harder to generalise classification boundaries as the nearest-neighbour distance between points becomes increasingly sparse making it difficult to correctly infer trends. We use a 4-dimensional dataset. However, we recognise that the effective input dimension of the STCA classifier is much greater.
- Permitting a large number of EA iterations: allowing an evolutionary algorithm to run for an excessive number of iterations ensures that the Pareto Front has fully converged and prevents ‘early stopping’ inhibiting the over-training of a classifier. We run our EA for 50,000 iterations.
- Increasing the complexity of the problem: to a lesser extent, the degree of overlap observed between our Gaussian distributions can influence over-fitting, as with few training data points it becomes harder to infer the correct distribution centers and locate the optimum separation boundary.

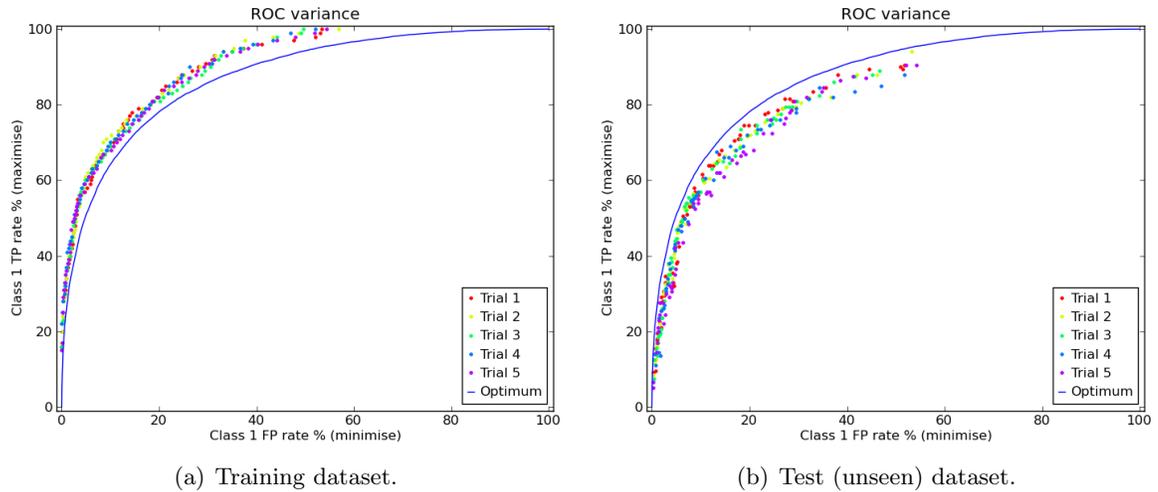


Figure 7.5: Single-archive neural network optimisation results (base-case). Coloured symbols indicate the different trail runs. The solid blue line indicates the Bayes optimal ROC curve.

Method	Training average	Diff. between optimum and training average	Test average	Diff. between optimum and test average	Diff. between training average and test average
Base-case	0.907	+0.033	0.801	-0.073	-0.106

Table 7.1: Comparison of AUCs. The optimum AUC is 0.874.

7.2.1 Single-archive trial (base-case)

Our base-case, to which all other trials will be compared, consists of running a single-archive optimiser without modification on the training dataset, then comparing the TP/FP rates of archive members when evaluated on the unseen test dataset. The area under the ROC curve (area under curve – AUC) is calculated for each run of the optimiser and averaged over the number of runs. Comparison to the AUC of the known ‘optimal’ ROC can then be made in order to assess classifier performance. We repeat all trials 5 times to get an idea of the variability between runs.

Ideally we would like the optimiser to generate a Pareto front that closely matches the optimal ROC and does not vary when applied to the unseen test dataset.

Base-case results

Referring to figure 7.5(a), as expected the optimiser has over-trained the neural network weights resulting in an estimated Pareto front that exceeds the optimal ROC in each trial. Evaluating members of the Pareto Front on test data results in TP/FP rates as seen in figure 7.5(b), these have varied significantly from the original training positions and now lie beneath the optimal ROC. A numerical summary of the corresponding AUCs may be seen in table 7.1. All subsequent trials will be compared to these figures and, we hope, will show an improvement over them in the consistency of performance between training and test dataset.

Cross-validation is often used to estimate how well a model will perform in practice (how well it will generalise to an ‘unseen’ dataset). The technique is designed to alleviate ‘Type III’ errors¹ in situations where simply increasing the size of the training dataset is prohibitive. In general the technique requires that a training set be divided at random into smaller subsets, the model is trained using one subset and validated using the those remaining. Validation results may be averaged in an effort to reduce variability and give a clearer picture as to the model’s likely actual performance. The following optimisation methods are all in some way based on this premise, we implement and test several distinct variations.

7.2.2 Two archive cross-validation with swap

Here we divide our original training dataset at random into two subsets (\mathcal{D}_1 and \mathcal{D}_2), each with an equal number of members and the same ratio of wanted to unwanted instances. In this implementation the optimiser is setup with two archives, one is designated as the ‘active’ archive (A), the other the ‘validation’ (V). The active archive behaves in the usual fashion, maintaining an elitist non-dominated set of parametrisations, these are evaluated on training subset \mathcal{D}_1 . Members of the validation archive are evaluated on the remaining subset \mathcal{D}_2 . However, in order for solutions to enter the validation archive they must first have been accepted into the active archive *and* have been found not to be dominated by the current members of the validation archive. Each time a new parametrisation enters the validation archive the archives swap roles. That is, the active archive now becomes the validation archive and the validation archive is designated as active. At completion of the optimisation run we take members of the current validation archive to be the optimised Pareto Front. Algorithm 14 outlines this procedure more formally.

This approach can be thought of as a modified form of ‘early stopping’ (see e.g. [10]); without the swapping of archive roles this scheme would be equivalent to ordinary early stopping. By implementing a second ‘validation’ archive we hope to better promote parametrisations that generalise well. Those that satisfy the ‘active’/training data subset provide the selective pressure, however the validation archive should act as a filter, ensuring that sufficiently generalised solutions are regularly re-introduced to the population each time the archive roles swap. That is, by alternating archive roles we aimed to re-introduce generalising parametrisations in the hope that they might procreate.

¹Type I error: rejecting the null hypothesis when it is true (False Positive).
Type II error: accepting the null hypothesis when it is false (True Positive).
Type III error: correctly rejecting the null hypothesis for the wrong reason (in the case of our classifier, learning the data inclusive of noise, not the overall trend(s)).

Algorithm 14 Two-archive swap Main()

Require: N	Number of iterations
Require: \mathcal{D}_1	Training data subset
Require: \mathcal{D}_2	Validation data subset
1: $A := \text{initialiseActive}()$	Initialise active archive A
2: $V := \text{initialiseValidation}(A)$	Initialise validation archive V
3: for $n := 1$ to N do	
4: $\theta := \text{select}(A)$	Select a parent, θ , to perturb
5: $\theta' := \text{peturb}(\theta)$	Perturb parent to create a new child
6: $(TP(\theta'), FP(\theta')) := \text{evaluate}(\theta', \mathcal{D}_1)$	Evaluate child's TP and FP rates on data subset \mathcal{D}_1
7: $accepted := \text{update}(A, \theta')$	Update archive, A , with child θ'
8: if $accepted$ then	
9: $(TP(\theta'), FP(\theta')) := \text{evaluate}(\theta', \mathcal{D}_2)$	Evaluate child's TP and FP rates on data subset \mathcal{D}_2
10: $\text{update}(V, \theta')$	Update archive, V , with child θ'
11: $T := A$	Swap archive roles
12: $A := V$	
13: $V := T$	
14: end if	
15: end for	Loop for N generations

Two-archive swap results

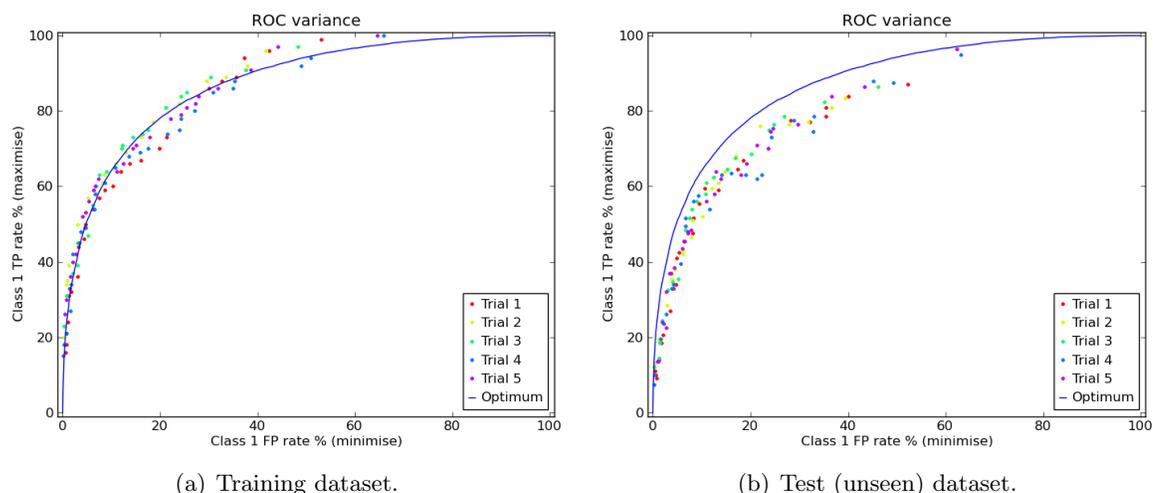


Figure 7.6: Two-archive swap neural network optimisation results. Coloured symbols indicate the different trial runs. The solid blue line indicates the Bayes optimal ROC curve.

Method	Training average	Diff. between optimum and training average	Test average	Diff. between optimum and test average	Diff. between training average and test average
Base-case	0.907	+0.033	0.801	-0.073	-0.106
Two-archive swap	0.863	-0.010	0.775	-0.098	-0.088

Table 7.2: Comparison of AUCs. The optimum AUC is 0.874.

Comparing figure 7.6(a) to figure 7.5(a) we can assert from visual inspection that the two-archive optimiser has not over-trained as much as in the base-case, with parametrisations located roughly over the optimal ROC ². However, the spread of solutions is much more pronounced between trials on the two-archive optimiser, implying that there may be more of a repeatability issue with convergence, particularly at higher TP/FP rates. Most disappointingly figure 7.6(b) shows that when evaluated on the test data there is still a considerable divergence of estimated Pareto solutions from the optimal ROC.

Table 7.2 confirms these findings. Compared to the base-case the two-archive swap results show that the average difference in AUC between the training data and the optimum is less (performance is closer to the expected optimum). However, while the base-case is shown to over-train by +0.033, the two-archive swap actually lies behind the optimum by -0.010, and has not converged enough. Two-archive swap appears to perform much worse when applied to the test dataset, the difference in performance between the optimum and the test average being -0.098 (compared to -0.073 in the base-case). In terms of the discrepancy in performance between the training and test datasets, the change is far smaller at -0.088 in the two-archive trials; however, the TP/FP rates on the training data were lower to start with.

²Note that for comparison purposes, parametrisations found in the final validation archive have been re-evaluated on the training dataset as a whole, and members subsequently found to be dominated have been removed.

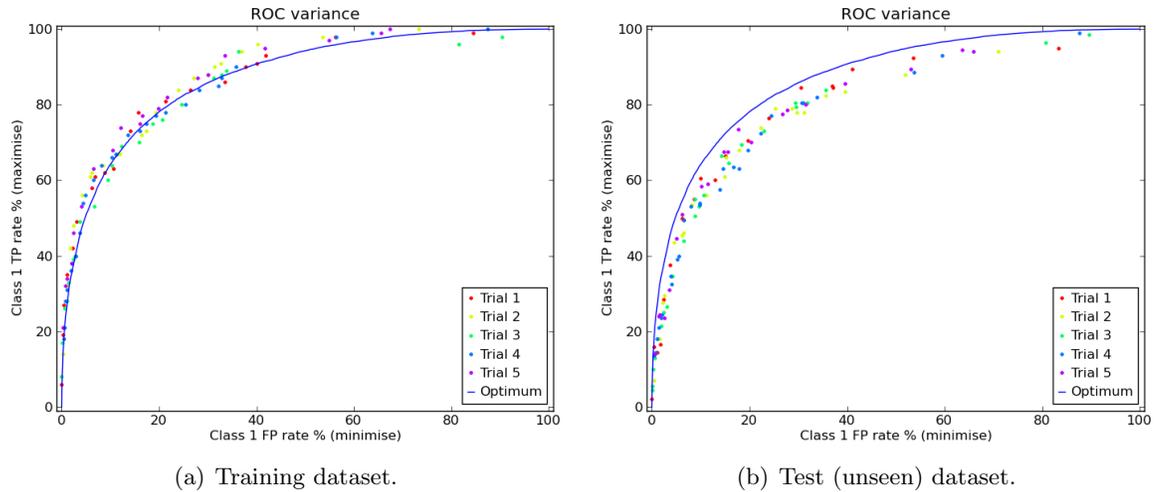


Figure 7.7: Two-archive no swap neural network optimisation results. Coloured symbols indicate the different trail runs. The solid blue line indicates the Bayes optimal ROC curve.

We cannot take only the difference in performance as a measure of success, the optimisation process must also result in a classifier that closely matches the optimum.

The average number of archive role swaps across all five trials was 81 (min. 67, max. 100), the majority of which occurred in the first $\approx 10,000$ iterations. This implies that the remaining 40,000 iterations contributed nothing to the final set of parametrisations (as we only consider those present in the designated ‘validation’ archive at the end of each run). It is possible that this swapping scheme has led to more erratic convergence. In fact it is feasible, after the first two swaps, that generalising/validated parametrisations may never get selected for perturbation. This can occur either because the proportion of ‘validated’ to previously ‘active’ members in the now active archive is low, reducing chances of selecting a ‘validated’ member, or because children of previously ‘active’ (non-validated) parametrisations are quickly found to dominate all ‘validated’ members that are present. Essentially the scheme does not reliably re-introduce generalising solutions for selection. There may be a benefit in evaluating solutions using a second validation dataset (to enforce early stopping), however the concept of swapping archive roles appears flawed.

7.2.3 Two archive cross-validation no swap

As in section 7.2.2 we suggested that the swapping of active and validation archive roles may be of little benefit, for comparison we will now re-run the scheme with archive swapping disabled. As before, in order for solutions to enter the validation archive they must first have been accepted into the active archive *and* must not be dominated by members of the validation archive. Lines 1 to 10 of algorithm 14 remain unchanged, lines 11, 12 and 13 are discarded.

Method	Training average	Diff. between optimum and training average	Test average	Diff. between optimum and test average	Diff. between training average and test average
Base-case	0.907	+0.033	0.801	-0.073	-0.106
Two-archive swap	0.863	-0.010	0.775	-0.098	-0.088
Two-archive no swap	0.871	-0.003	0.797	-0.076	-0.073

Table 7.3: Comparison of AUCs. The optimum AUC is 0.874.

Two-archive no swap results

Referring to figure 7.7(a), the training fronts have done very well at matching the optimal ROC, the average AUC is only -0.003 away from the optimum (table 7.3) and the optimiser has managed to locate Pareto members that reach into higher TP/FP regions of the curve. However, performance on the test data is worse than our base-case (although better than two-archive with swap) and, as illustrated in figure 7.7(b), re-evaluation of members on the test dataset shows them to be sub-optimal. Two-archive (no swap) has shown a slight improvement over the base-case in reducing over-training, co-incidentally increasing performance consistency; the difference between training and test AUC being -0.073 compared to -0.106 in the base-case.

Algorithm 15 Three-archive no swap Main()

Require: N	Number of iterations
Require: \mathcal{D}_1	Training data subset
Require: \mathcal{D}_2	Validation data subset 1
Require: \mathcal{D}_3	Validation data subset 2
1: $A := \text{initialiseActive}()$	Initialise active archive A
2: $V_1 := \text{initialiseValidation}(A)$	Initialise validation archive V_1
3: $V_2 := \text{initialiseValidation}(V_1)$	Initialise validation archive V_2
4: for $n := 1$ to N do	
5: $\theta := \text{select}(A)$	Select a parent, θ , to perturb
6: $\theta' := \text{perturb}(\theta)$	Perturb parent to create a new child
7: $(TP(\theta'), FP(\theta')) := \text{evaluate}(\theta', \mathcal{D}_1)$	Evaluate child's TP and FP rates on data subset \mathcal{D}_1
8: $accepted1 := \text{update}(A, \theta')$	Update archive, A , with child θ'
9: if $accepted1$ then	
10: $(TP(\theta'), FP(\theta')) := \text{evaluate}(\theta', \mathcal{D}_2)$	Evaluate child's TP and FP rates on data subset \mathcal{D}_2
11: $accepted2 := \text{update}(V_1, \theta')$	Update archive, V_1 , with child θ'
12: if $accepted2$ then	
13: $(TP(\theta'), FP(\theta')) := \text{evaluate}(\theta', \mathcal{D}_3)$	Evaluate child's TP and FP rates on data subset \mathcal{D}_3
14: $\text{update}(V_2, \theta')$	Update archive, V_2 , with child θ'
15: end if	
16: end if	
17: end for	Loop for N generations

7.2.4 Three archive cross-validation no swap

In the interests of completeness, and as two-archive (no swap) is shown to be of some benefit, we now increase the number of validation data subsets from one to two. Similar to the previous implementation, we divide our original training dataset at random into three subsets (\mathcal{D}_1 , \mathcal{D}_2 and \mathcal{D}_3), each with an approximately equal number of members and the same ratio of wanted to unwanted instances. We maintain three archives, an ‘active’ one (A) and two validation archives (V_1 and V_2), members of these archives are evaluated on data subsets \mathcal{D}_1 , \mathcal{D}_2 and \mathcal{D}_3 respectively. Solutions entering validation archive V_1 must first have been accepted into the active archive and have been found not to be dominated by members of V_1 . Solutions entering validation archive V_2 must first have been accepted into validation archive V_1 and have been found not to be dominated by members of V_2 . This procedure is outlined in algorithm 15. Once again, only members present in the final validation archive (V_2) are considered when optimisation completes. Essentially we just add an additional validation filter to help ensure that parametrisations are sufficiently generalised.

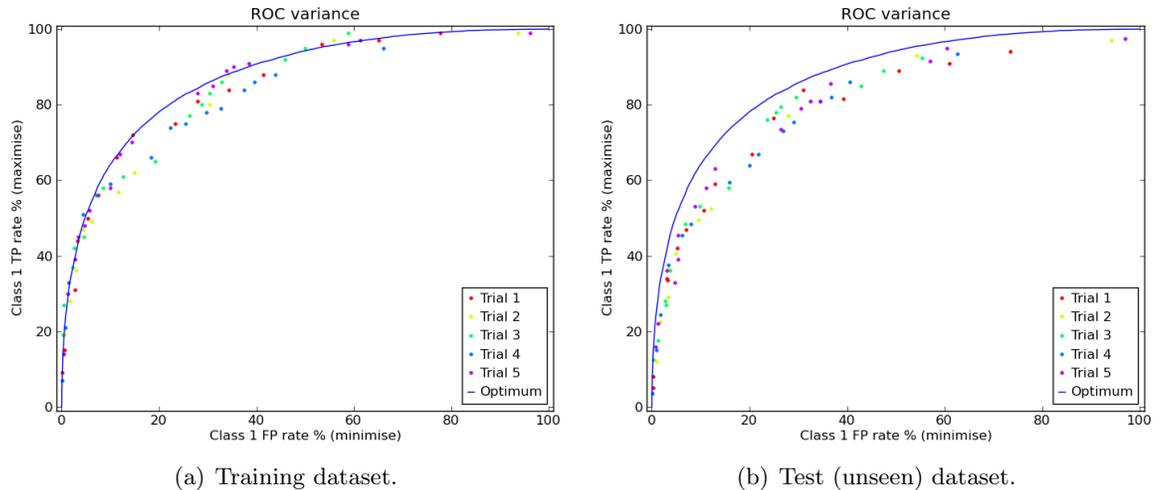


Figure 7.8: Three-archive no swap neural network optimisation results. Coloured symbols indicate the different trail runs. The solid blue line indicates the Bayes optimal ROC curve.

Method	Training average	Diff. between optimum and training average	Test average	Diff. between optimum and test average	Diff. between training average and test average
Base-case	0.907	+0.033	0.801	-0.073	-0.106
Two-archive swap	0.863	-0.010	0.775	-0.098	-0.088
Two-archive no swap	0.871	-0.003	0.797	-0.076	-0.073
Three-archive no swap	0.826	-0.047	0.783	-0.091	-0.043

Table 7.4: Comparison of AUCs. The optimum AUC is 0.874.

Three-archive no swap results

Figure 7.8(a) shows that the training archive is now under-training, with the majority of parametrisations being situated below the optimal ROC. There is also a higher degree of variance between trials. However, the Pareto front has managed to extend further still into the higher TP/FP rates, nearing 100% on both objectives. Table 7.4 highlights that on both the training and the test datasets the three-archive optimiser has performed worse than the base-case. The difference between the training and test average AUC is much less than previous trials at -0.043 , meaning performance between datasets is the most consistent seen so far, however this is because the optimiser has not fully converged on the training data and is sub-optimal in both cases. This lack of convergence is most likely due to parametrisations no longer entering validation archive V_2 after the first few thousand iterations. As the optimiser enters a ‘refinement’ phase, where it makes smaller adjustments to members of the active archive that push the corresponding estimated Pareto front towards its final position, parametrisations present in archives A and V_1 are no longer sufficiently similar to those in V_2 . This means that there is a genetic disparity between those parametrisations in A undergoing selection and perturbation, and those configurations that might dominate members of V_2 , meaning development stops too early leaving under-trained solutions.

7.2.5 Bootstrap evaluation

To address the issue of genetic divergence between active and validation archives raised in section 7.2.4, we now attempt to implement validation of parametrisations on a finer, individual, rather than population based, scale. The term ‘bootstrapping’ is used to refer to a specific type of data sampling [26]. Unlike cross-validation, which focuses on the repeated use of predetermined (fixed) data subsets, bootstrapping refers to the repeated sampling of the data; each bootstrap sample can be viewed as a possible statistically equivalent realisation of the data [26].

The goal is to ascertain the worst possible TP/FP rate that we can expect of each parametrisation and use this as a guide to optimisation. Initially we might expect that this process will push the front forwards in the usual manner, simply using an alternative reference point (the worst TP/FP rates rather than the average). However, as convergence stabilises, in order to push any further forwards, the variance of members on data re-samples must be addressed. That is, the best ‘average’ position in TP/FP space may have been reached but the expected worst TP/FP rates can still be improved by addressing the variance. We hope that over many iterations the expected worst performance of members will improve as configurations that are more resilient to variance, i.e. those that generalise well, are promoted.

To implement bootstrapping a random sample of instances are taken from the training dataset, with replacement permitted (i.e. there may be duplicates of a given instance present in the sampled dataset). The size of the final sampled dataset will equal that of the original dataset, however the contents will vary. We amend the optimiser’s evaluation function such that each new parametrisation is evaluated on multiple bootstrap-samples and the resulting TP/FP rates for each recorded. We can then calculate the *worst* TP and FP rates observed and use this for fitness comparison. Note that these are the worst TP and FP rates taken separately over the evaluations, not the worst single instance.

It should be noted that averaging multiple re-sampled rates together should equate to the TP/FP rates of the parametrisation if it were evaluated on the original un-sampled training dataset (provided that enough bootstrap-samples are taken). Thus, in actuality, we need not re-evaluate each data bootstrap-sample on the classifier, but instead we need only evaluate the training dataset as a whole and then sample the resulting classifications themselves in order to obtain the re-calculated TP/FP rates. This is a useful feature in expensive optimisation problems as the re-sampling can be performed ‘off-line’.

In our trials we re-sample the training data 20 times per evaluation. However, to ensure that the ‘worst’ TP/FP rates sampled are never better than the average TP/FP rates, which may occur in situations where a restricted number of bootstrap-samples are taken, we initialise $worstTP := averageTP$ and $worstFP := averageFP$. An alternative to guard against this eventuality would, of course, be to sufficiently increase the number of bootstrap-samples taken, however this would lead to an increase in processing time; even with only 20 samples we are effectively processing $20 \times 50,000$ iterations (1,000,000) which takes a comparatively long time, regardless of off-line re-sampling.

In order to integrate bootstrapping into the single-archive optimiser the `evaluate()`

Algorithm 16 $\text{evaluate}(\theta')$

Require: \mathcal{D}	Training dataset \mathcal{D}
Require: \mathcal{L}	Class labels corresponding to dataset \mathcal{D} where label \mathcal{L}_i corresponds to data instance \mathcal{D}_i
Require: S	Number of re-samples to perform
1: $\mathcal{R} := \text{classifier}(\theta', \mathcal{D})$	Classify dataset \mathcal{D} , assign results to \mathcal{R} where \mathcal{R}_i corresponds to data instance \mathcal{D}_i
2: $\text{worstTP} := \text{calcTP}(\mathcal{L}, \mathcal{R})$	Initialise as ‘average’ TP rate
3: $\text{worstFP} := \text{calcFP}(\mathcal{L}, \mathcal{R})$	Initialise as ‘average’ FP rate
4: for $n := 1$ to S do	
5: $(TP, FP) := \text{resample}(\mathcal{L}, \mathcal{R})$	Perform re-sampling, see algorithm 17
6: if $TP < \text{worstTP}$ then	
7: $\text{worstTP} := TP$	Record worst TP rate observed so far
8: end if	
9: if $FP > \text{worstFP}$ then	
10: $\text{worstFP} := FP$	Record worst FP rate observed so far
11: end if	
12: end for	Loop for S re-samples
13: return $(\text{worstTP}, \text{worstFP})$	Return the worst TP/FP rates observed

Algorithm 17 $\text{resample}(\mathcal{L}, \mathcal{R})$

1: $\mathcal{S}_{\mathcal{L}} := \emptyset$	Initialise re-sampled set of correct labels $\mathcal{S}_{\mathcal{L}}$ as an empty set
2: $\mathcal{S}_{\mathcal{R}} := \emptyset$	Initialise re-sampled set of results $\mathcal{S}_{\mathcal{R}}$ as an empty set
3: for $n := 1$ to $\text{len}(\mathcal{L})$ do	
4: $i := \text{rand}(0, \text{len}(\mathcal{L}))$	Generate a random index within range of the dataset, duplicate instances are permitted
5: $\mathcal{S}_{\mathcal{L}} := \text{append}(\mathcal{S}_{\mathcal{L}}, \mathcal{L}_i)$	Append the sampled label, \mathcal{L}_i , to the re-sampled labels subset, $\mathcal{S}_{\mathcal{L}}$
6: $\mathcal{S}_{\mathcal{R}} := \text{append}(\mathcal{S}_{\mathcal{R}}, \mathcal{R}_i)$	Append the sampled result, \mathcal{R}_i , to the re-sampled results subset, $\mathcal{S}_{\mathcal{R}}$
7: end for	
8: $TP := \text{calcTP}(\mathcal{S}_{\mathcal{L}}, \mathcal{S}_{\mathcal{R}})$	Calculate the TP rate of the re-sampled data
9: $FP := \text{calcFP}(\mathcal{S}_{\mathcal{L}}, \mathcal{S}_{\mathcal{R}})$	Calculate the FP rate of the re-sampled data
10: return (TP, FP)	Return the TP/FP rates of the re-sampled data

function must be amended as outlined in algorithm 16. At line 1 the *whole* training dataset, \mathcal{D} , is evaluated on the new classifier, θ' (the neural network), the output of which is an array of class decisions, \mathcal{R} . Lines 2 and 3 calculate and assign the TP/FP rate of the classifications as the initial ‘worst’ values (these correspond to the expected average TP/FP rate of the classifier on the dataset). Line 5 the performs the actual sub-sampling of the evaluated results (see algorithm 17), and lines 6 to 11 determine whether a new worst-case TP or FP rate has been found. At line 13 the worst observed TP/FP rates are returned as the result of classifier evaluation on the dataset. Note that the `evaluate()` function really returns $\min(\text{averageTPFP}, \text{worstSampledTPFP})$.

Algorithm 17 repeatedly samples members of the training results set. At lines 1 and 2, new dataset label, $\mathcal{S}_{\mathcal{L}}$, and result, $\mathcal{S}_{\mathcal{R}}$, instance sets are initialised. Between lines 3 and 7 classified instances, \mathcal{R}_i , and their corresponding correct class labels, \mathcal{L}_i , are randomly (with duplication) appended to the sets. The new TP/FP rates for the sub-sampled dataset are then calculated (lines 8 and 9) and returned (line 10).

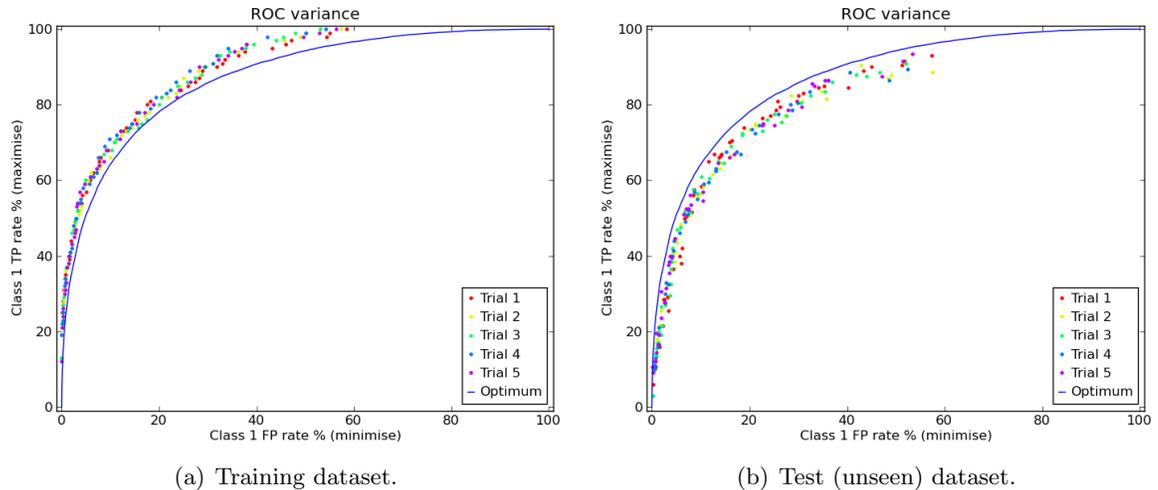


Figure 7.9: Bootstrap evaluation, neural network optimisation results. Coloured symbols indicate the different trail runs. The solid blue line indicates the Bayes optimal ROC curve.

Method	Training average	Diff. between optimum and training average	Test average	Diff. between optimum and test average	Diff. between training average and test average
Base-case	0.907	+0.033	0.801	-0.073	-0.106
Two-archive swap	0.863	-0.010	0.775	-0.098	-0.088
Two-archive no swap	0.871	-0.003	0.797	-0.076	-0.073
Three-archive no swap	0.826	-0.047	0.783	-0.091	-0.043
Bootstrap	0.901	+0.027	0.806	-0.067	-0.094

Table 7.5: Comparison of AUCs. The optimum AUC is 0.874.

Bootstrap evaluation results

Figure 7.9(a) clearly indicates that the classifier is once again over-fitting, although as table 7.5 illustrates, not as much as the base-case. On the test data (figure 7.9(b)), classifier performance is the best observed thus far at -0.067 from the optimum. Due to over-training, in terms of performance consistency between training and test data the bootstrapping optimiser is second to last, bettering the base-case only, however, as long as the test-case performance is good (which it is), the difference between training and test averages is of little importance.

Comparing worst TP/FP to average TP/FP

In the previous set of trials we compared the (estimated) worst TP/FP rates of each given parametrisation when determining dominance. The hope was that, by optimising based on worst case results, we might promote members of the population that generalise better as, over many iterations, the variance must decrease in order to push the front forwards. In this section we experiment with comparing the worst TP/FP rates of new parametrisations, potentially entering the archive, to the average TP/FP of members already in the archive. This means that the expected worst TP/FP rates of new parametrisations really do have to be

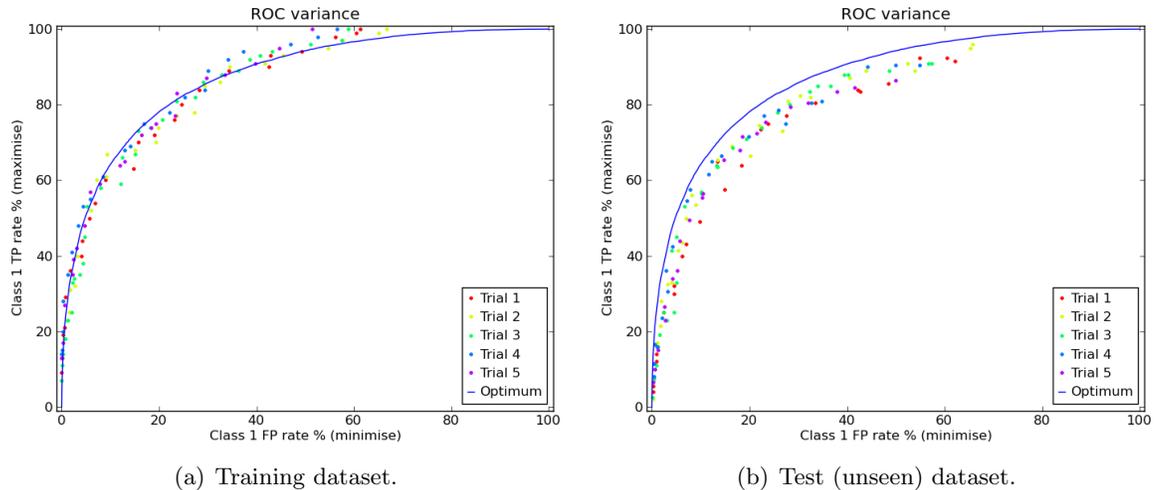


Figure 7.10: Bootstrap evaluation (worst–average), neural network optimisation results. Coloured symbols indicate the different trail runs. The solid blue line indicates the Bayes optimal ROC curve.

Method	Training average	Diff. between optimum and training average	Test average	Diff. between optimum and test average	Diff. between training average and test average
Base-case	0.907	+0.033	0.801	−0.073	−0.106
Two-archive swap	0.863	−0.010	0.775	−0.098	−0.088
Two-archive no swap	0.871	−0.003	0.797	−0.076	−0.073
Three-archive no swap	0.826	−0.047	0.783	−0.091	−0.043
Bootstrap (worst–worst)	0.901	+0.027	0.806	−0.067	−0.094
Bootstrap (worst–average)	0.868	−0.006	0.793	−0.080	−0.074

Table 7.6: Comparison of AUCs. The optimum AUC is 0.874.

substantially better those members already present in the archive in order to enter, effectively discarding smaller incremental changes that ‘fine-tune’ the front to the dataset (inclusive of noise) and only permitting larger movements in objective space, as these are more likely to be representative of having ‘learned’ something beneficial about data trends.

Results of worst–average comparison method

Referring to figure 7.10(a), as expected this method of comparison has indeed reduced overfitting to the training data, resulting in a training AUC very close to the optimum, the difference being -0.006 (see table 7.6). However, as figure 7.10(b) illustrates, evaluation on the test dataset is less successful. The location of the Pareto Fronts is far less consistent between trials (compared to the original bootstrap implementation), resulting in a disappointing average AUC that is -0.080 from the optimum.

Overall, this implementation appears to generate results roughly equivalent to the two-archive no swap, optimisation method. However, two-archive no swap betters it (just) across all measures and is computationally cheaper.

Algorithm 18 $\text{resample}(\mathcal{L}, \mathcal{R})$

1: $\mathcal{S}_{\mathcal{L}} := \emptyset$	Initialise re-sampled set of correct labels $\mathcal{S}_{\mathcal{L}}$ as an empty set
2: $\mathcal{S}_{\mathcal{R}} := \emptyset$	Initialise re-sampled set of results $\mathcal{S}_{\mathcal{R}}$ as an empty set
3: $(\mathcal{L}, \mathcal{R}) := \text{shuffle}(\mathcal{L}, \mathcal{R})$	Shuffle members of the classified dataset, this ensures that duplicate instances are not permitted
4: for $i := 1$ to $\text{len}(\mathcal{L})/2$ do	
5: $\mathcal{S}_{\mathcal{L}} := \text{append}(\mathcal{S}_{\mathcal{L}}, \mathcal{L}_i)$	Append the first half of the shuffled labels in \mathcal{L} to the re-sampled labels subset, $\mathcal{S}_{\mathcal{L}}$
6: $\mathcal{S}_{\mathcal{R}} := \text{append}(\mathcal{S}_{\mathcal{R}}, \mathcal{R}_i)$	Append the first half of the shuffled results in \mathcal{R} to the re-sampled results subset, $\mathcal{S}_{\mathcal{R}}$
7: end for	
8: $TP := \text{calcTP}(\mathcal{S}_{\mathcal{L}}, \mathcal{S}_{\mathcal{R}})$	Calculate the TP rate of the re-sampled data
9: $FP := \text{calcFP}(\mathcal{S}_{\mathcal{L}}, \mathcal{S}_{\mathcal{R}})$	Calculate the FP rate of the re-sampled data
10: return (TP, FP)	Return the TP/FP rates of the re-sampled data

7.2.6 Randomise evaluation

In the previous section we assessed the effect of re-sampling data (with replacement permitted) on resulting classifier variance. For the sake of completeness we run a series of similar trials, this time *without* permitting replacements in the re-sampled dataset. That is, a randomised subset of the training data will be evaluated each time. The size of the final re-sampled dataset is smaller than that of the original dataset (50% smaller in our implementation), specific instances contained within will vary and no single instance will be duplicated. All that is required to integrate these changes into the optimiser is to replace algorithm 17 with the implementation outlined in algorithm 18.

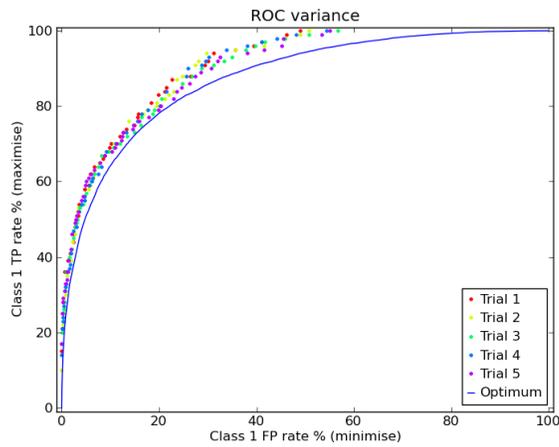
At lines 1 and 2, new dataset label, $\mathcal{S}_{\mathcal{L}}$, and result, $\mathcal{S}_{\mathcal{R}}$, instance sets are initialised. At line 3 members of the evaluated instances set and their corresponding correct class labels are ‘shuffled’ (re-ordered at random). Lines 4 to 7 then copy the first half of the re-shuffled instances into sets $\mathcal{S}_{\mathcal{L}}$ and $\mathcal{S}_{\mathcal{R}}$. The new TP/FP rates for the re-sampled dataset are then calculated (lines 8 and 9) and returned (line 10).

Randomise evaluation results

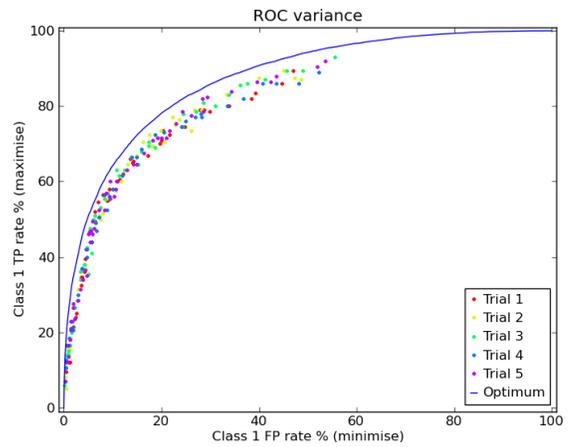
Note that for completeness, we have included results for both previously mentioned TP/FP rate comparison methods; worst–worst, and worst–average.

Referring to table 7.7 (Randomise, worst–worst); unfortunately this approach to optimisation appears to generate results roughly equivalent to the base-case. On the training data over-fitting is clearly occurring (figure 7.11(a)), resulting in sub-optimal results on the test data (figure 7.11(b)).

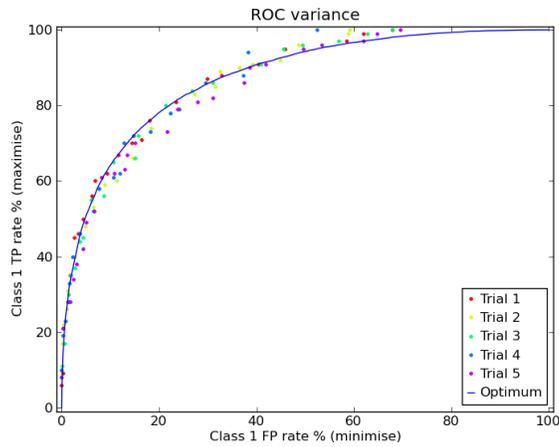
When using the estimated worst TP/FP vs. average TP/FP comparison method, results do improve. Similar to the bootstrap trials, the use of comparing parametrisations for dominance based on the worst–average method appears to be successful at inhibiting over-fitting of the training dataset, producing an average AUC closer to that of the optimal ROC (see figure 7.11(c) / table 7.7). However, it would appear that the bootstrap method of re-sampling is far more effective at reducing variance in both cases.



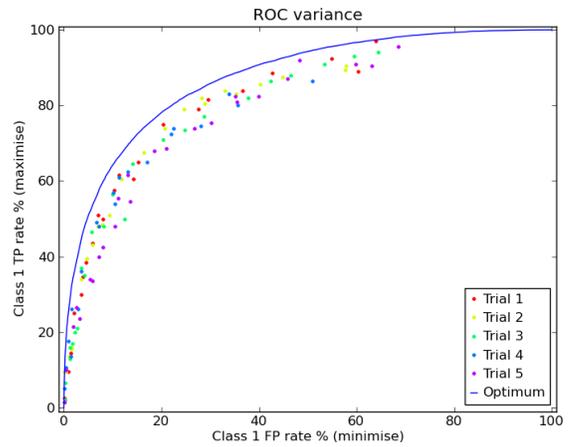
(a) Training dataset (worst-worst).



(b) Test (unseen) dataset (worst-worst).



(c) Training dataset (worst-average).



(d) Test (unseen) dataset (worst-average).

Figure 7.11: Randomise evaluation, neural network optimisation results. Coloured symbols indicate the different trail runs. The solid blue line indicates the Bayes optimal ROC curve.

Method	Training average	Diff. between optimum and training average	Test average	Diff. between optimum and test average	Diff. between training average and test average
Base-case	0.907	+0.033	0.801	-0.073	-0.106
Two-archive swap	0.863	-0.010	0.775	-0.098	-0.088
Two-archive no swap	0.871	-0.003	0.797	-0.076	-0.073
Three-archive no swap	0.826	-0.047	0.783	-0.091	-0.043
Bootstrap (worst-worst)	0.901	+0.027	0.806	-0.067	-0.094
Bootstrap (worst-average)	0.868	-0.006	0.793	-0.080	-0.074
Randomise (worst-worst)	0.903	+0.030	0.800	-0.074	-0.103
Randomise (worst-average)	0.864	-0.010	0.793	-0.081	-0.071

Table 7.7: Comparison of AUCs. The optimum AUC is 0.874.

7.2.7 Probabilistic archive

Our final method of reducing classifier variance is based on the works of [77, 54, 38, 39], and their proposals for probabilistic based dominance.

As previously stated at the beginning of section 7.1, perhaps the easiest solution to overfitting is to provide a larger training dataset that is better representative of broader trends to be modeled. This rhetoric is based on the fact that for any finite dataset there is an associated uncertainty in ascertaining the operating point, that can only be reduced at the expense of evaluating more data. As illustrated by our randomised evaluation trials, a repeated draw of the same size from the same data will often result in a different objective evaluation. This is true for all finite data, although the uncertainty (measured by the variance) in the mean reduces with the square root of the number of data samples [39, 25].

This leads to a potential problem in dominance based comparisons. Parametrisations may be prohibited from entering an archive if they are deemed to be dominated by current members, when in fact this may not be the case – noise in the data has made them seem worse than they really are. Conversely, current members of the archive that should not be present, may be – noise has resulted in these being given undue significance. Finally, as is the case with the STCA parametrisations shown in figure 7.1, the optimised Pareto front may over-estimate the true Pareto front, giving an unrealistic representation of classifier performance [38].

Ultimately, the uncertainty associated to objective evaluations may lead to a reduction in algorithm efficiency. The premise behind the ‘probabilistic archive’ is to maintain a set of parametrisations that are *probably* not dominated by any other member, for which a degree of confidence may be specified. The notion of dominance specified in section 2.4.3, can thus be modified as follows:

$$\boldsymbol{\theta} \prec^\alpha \boldsymbol{\phi} \Leftrightarrow p(\boldsymbol{\theta} \prec \boldsymbol{\phi}) \geq \alpha \quad (7.11)$$

When $\alpha = 1$ this reverts to the original deterministic dominance [39].

This new definition of probabilistic dominance will inherently generate a thicker archive of candidate parametrisations on which to perform optimisation. The confidence value, α , relates to how certain we want to be that the optimisation has not rejected parametrisations from the archive that may actually have dominated other members [39]. Thus hopefully maintaining parametrisations that are insensitive to noise and generalise well.

Referring to [39], the probability of dominance is calculated as follows: If the uncertainty affecting each objective’s evaluation can be assumed to be independent then the probability of dominance can be formulated as a product of probabilities for each objective dimension:

$$p(\boldsymbol{\theta} \prec \boldsymbol{\phi}) = \prod_{i=1}^D p(f_i(\boldsymbol{\theta}) < f_i(\boldsymbol{\phi})) \quad (7.12)$$

Here D is the number of objectives and f_i corresponds to the objective function for objective i . Note to simplify the notation we assume that we are *minimising* all objective functions (so

to turn maximisation of TP rate into a minimisation problem we must first invert the rate e.g. $1 - TPR$). If the errors are assumed to be Gaussian distributed the probability in terms of the error function is then:

$$p(f_i(\boldsymbol{\theta}) < f_i(\boldsymbol{\phi})) = \frac{1}{2} \left[1 + \operatorname{erf}(m/\sqrt{2}) \right] \quad (7.13)$$

where

$$m = \frac{f_i(\boldsymbol{\phi}) - f_i(\boldsymbol{\theta})}{\sqrt{\sigma_{f_i(\boldsymbol{\theta})}^2 + \sigma_{f_i(\boldsymbol{\phi})}^2}} \quad (7.14)$$

and $\sigma_{f_i(\boldsymbol{\theta})}^2$ is the variance in an objective's classification *rate* for parameter $\boldsymbol{\theta}$ [54]. Fieldsend and Everson show that the errors are will approximated by a Gaussian distribution when the number of positive or negative instances is greater than about 20 [39]. In this case the variance is

$$\sigma_{f_i(\boldsymbol{\theta})}^2 = \frac{f_i(\boldsymbol{\theta})(1 - f_i(\boldsymbol{\theta}))}{N_c} \quad (7.15)$$

where N_c corresponds to the number of a specific class instance as found in the dataset (i.e., the total number of wanted class instances found in the dataset or the total number of unwanted class instances found in the dataset, depending on the objective evaluated). Thus the probability of one solution dominating another can be calculated with increasing accuracy as the number of instances in the data increases (as the $\sigma_{f_i(\boldsymbol{\theta})}^2$ terms decrease). Note, if $f_i(\boldsymbol{\theta}) = f_i(\boldsymbol{\phi})$ then $m = 0$ and $p(f_i(\boldsymbol{\theta}) < f_i(\boldsymbol{\phi})) = \frac{1}{2}$, as expected [39].

Although computationally more expensive, in the case of STCA, the time taken to calculate probabilistic dominance is still relatively insignificant compared to the time taken to evaluate parametrisations on the data (the effect is even more negligible if some sort of parallelised evaluation scheme is implemented).

The final archive after optimisation finishes will contain many, potentially dominated, parametrisations. In order to select and generate an estimated Pareto front from those that are most generalising we use cross-validation. That is, the training dataset is divided at random into subsets \mathcal{D}_1 and \mathcal{D}_2 (the same as used in the two-archive trials). Optimisation using the probabilistic archive is carried out on subset \mathcal{D}_1 . Upon completion, the entire archive is re-evaluated on validation subset \mathcal{D}_2 , this time using a deterministic (non-probabilistic) dominance comparison. This generates a estimated Pareto front of non-dominant members that can be compared to previous trials.

In the following trials the confidence value, α , is not tuned however, three settings are tested in order to ascertain a rough perception of its effect on optimisation. It is probable that the optimum setting for α is problem specific. A value that is too large will result in an unnecessarily large archive (because more parametrisations are considered *probably* non-dominating), causing a drop in selection pressure that leads to slow convergence. Conversely, a small α value may undermine the benefits of maintaining a probabilistic archive, retaining

Method	Training average	Diff. between optimum and training average	Test average	Diff. between optimum and test average	Diff. between training average and test average
Base-case	0.907	+0.033	0.801	-0.073	-0.106
Two-archive swap	0.863	-0.010	0.775	-0.098	-0.088
Two-archive no swap	0.871	-0.003	0.797	-0.076	-0.073
Three-archive no swap	0.826	-0.047	0.783	-0.091	-0.043
Bootstrap (worst-worst)	0.901	+0.027	0.806	-0.067	-0.094
Bootstrap (worst-average)	0.868	-0.006	0.793	-0.080	-0.074
Randomise (worst-worst)	0.903	+0.030	0.800	-0.074	-0.103
Randomise (worst-average)	0.864	-0.010	0.793	-0.081	-0.071
Probabilistic archive 0.7	0.877	+0.003	0.804	-0.070	-0.073
Probabilistic archive 0.8	0.875	+0.001	0.809	-0.064	-0.065
Probabilistic archive 0.9	0.862	-0.012	0.808	-0.066	-0.054

Table 7.8: Comparison of AUCs. The optimum AUC is 0.874.

too few parametrisations (because more are considered *probably* dominating, causing members to be removed) and permitting over-training.

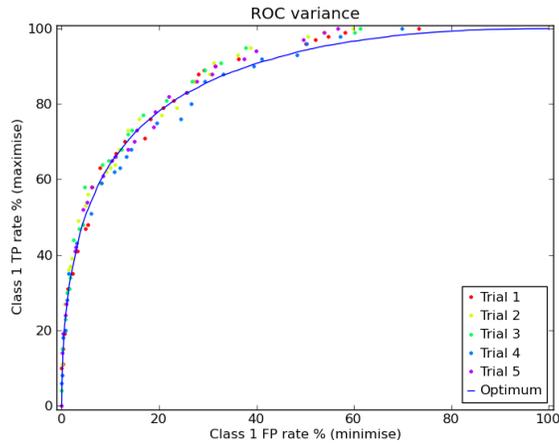
Probabilistic archive results

Referring to figure 7.12, it is difficult to visually compare what effect adjusting α has on optimisation; on the training data the trials appear to be centred around the optimal ROC and on test data all three α settings seem roughly comparable. However, table 7.8 proves more useful. The first thing to note is that, regardless of the value of α , the probabilistic archive is very good at matching the optimal AUC in training, particularly at 0.8, where the average AUC is only +0.001 from the optimum; referring to 7.12(c) we can see that members of the Pareto front are centred over the optimal ROC up to $\approx 60\%$ TP and that there is some slight over-training apparent from $\approx 90\%$ TP onwards.

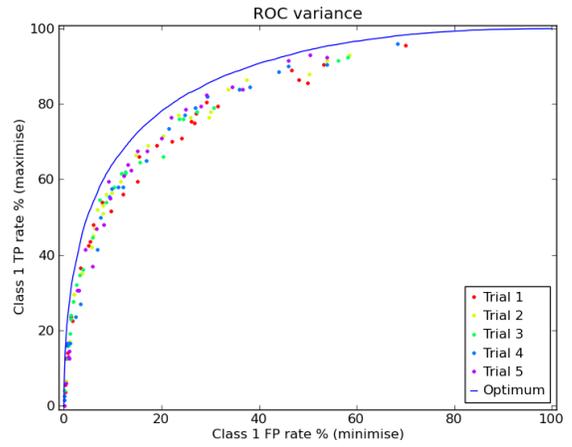
On the test data the probabilistic trials fare well too. When α is 0.8 and 0.9 the divergence from the optimal ROC is only -0.064 and -0.066 respectively, bettering all previous methods. When α equals 0.7 the probabilistic archive does less well, the divergence from the optimal ROC being -0.070. This is perhaps due to the probabilistic archive being too thin (see figure 7.13(a)) however, it is still bettered only by Bootstrap (worst-worst).

As expected, setting α too high begins to rapidly affect convergence. On the training data when α is set comparatively low at 0.7 and 0.8, the resulting divergence from the optimal AUC is +0.003 and +0.001 respectively (only very slight over-training). Whilst setting α too high at 0.9 results in a divergence from the optimal AUC of -0.012, a significant amount of under-fitting considering how close to the optimum the former divergence figures are.

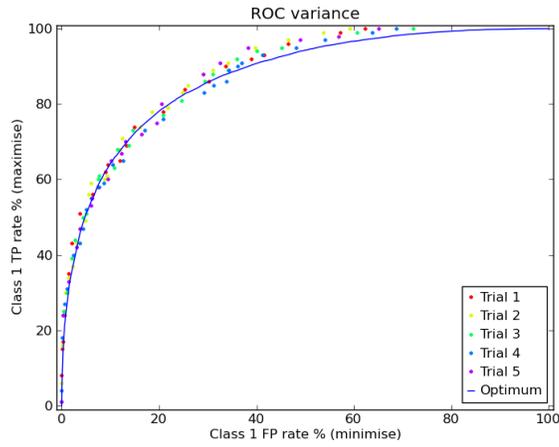
Finally, referring to figure 7.13, it is worth noting that without re-evaluating parametrisations on a second validation dataset, there is no easy way of predicting test performance from the location of members in the probabilistic archive.



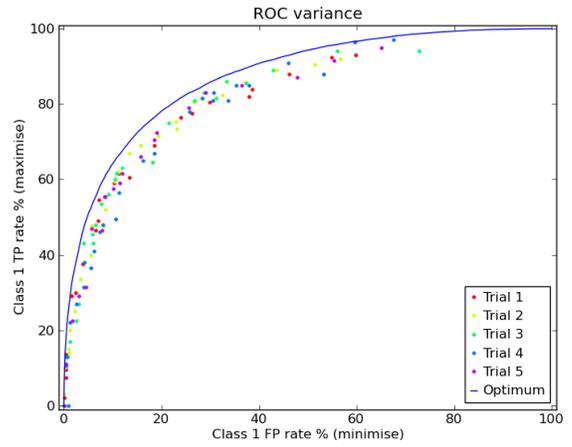
(a) Training dataset ($\alpha = 0.7$).



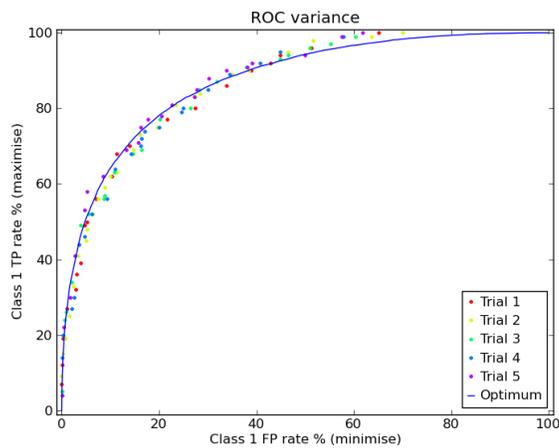
(b) Test (unseen) dataset ($\alpha = 0.7$).



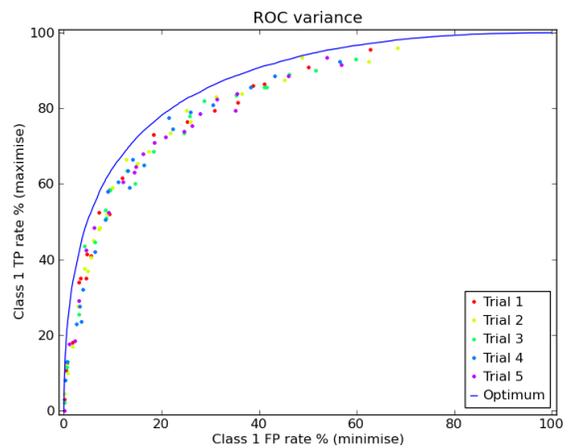
(c) Training dataset ($\alpha = 0.8$).



(d) Test (unseen) dataset ($\alpha = 0.8$).



(e) Training dataset ($\alpha = 0.9$).



(f) Test (unseen) dataset ($\alpha = 0.9$).

Figure 7.12: Probabilistic archive, neural network optimisation results. Coloured symbols indicate the different trail runs. The solid blue line indicates the Bayes optimal ROC curve.

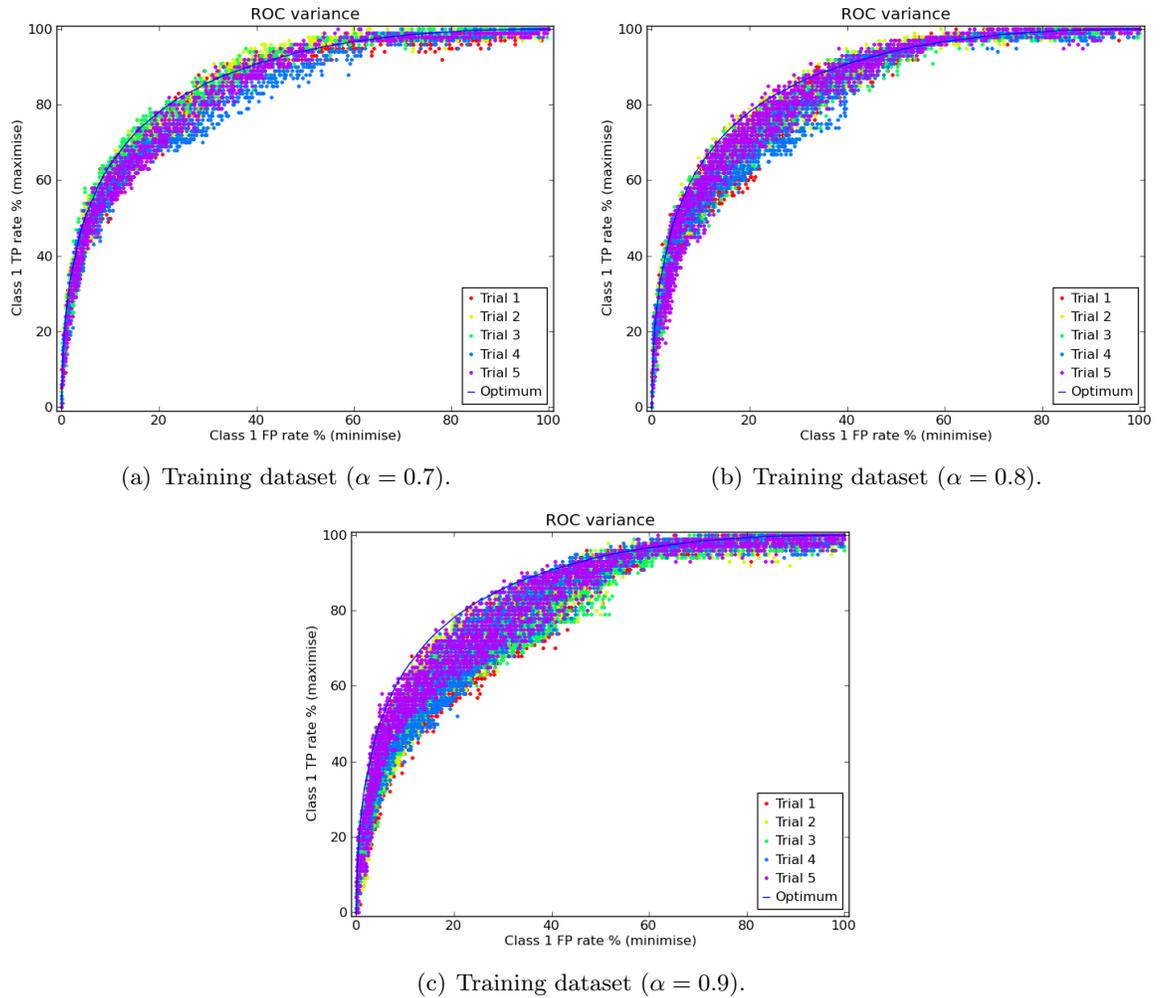


Figure 7.13: Probabilistic archive, contents after 50,000 iterations. Coloured symbols indicate the different trail runs. The solid blue line indicates the Bayes optimal ROC curve.

7.2.8 Summary of over-fitting mitigation methods

In the prior text we have presented seven distinct approaches to reducing classifier over-fitting when training a classifier using an EA. Using a ‘toy’ problem has permitted us to calculate a known optimal ROC for the classification task and we have taken the average Area Under Curve (AUC) as a measure of classifier performance in each instance. A final summary of results may be observed in table 7.8, page 151. In each example we compare the efficacy of the implementation to results from a single-archive elitist EA (our base-case).

In all but one instance (Three-archive no swap) the base-case was bettered by the techniques examined. That is, when training the difference between average AUC and the optimal AUC was reduced. Using a probabilistic archive with $\alpha = 0.8$ was best, almost matching the optimal AUC at 0.875, +0.001 from the optimum. When archive members were evaluated on the test data Two-archive swap, Two-archive no swap, Three-archive no swap, Bootstrap (worst-average), Randomise (worst-worst) and Randomise (worst-average) all performed worse than the base-case, increasing under-fitting. Bootstrap (worst-worst),

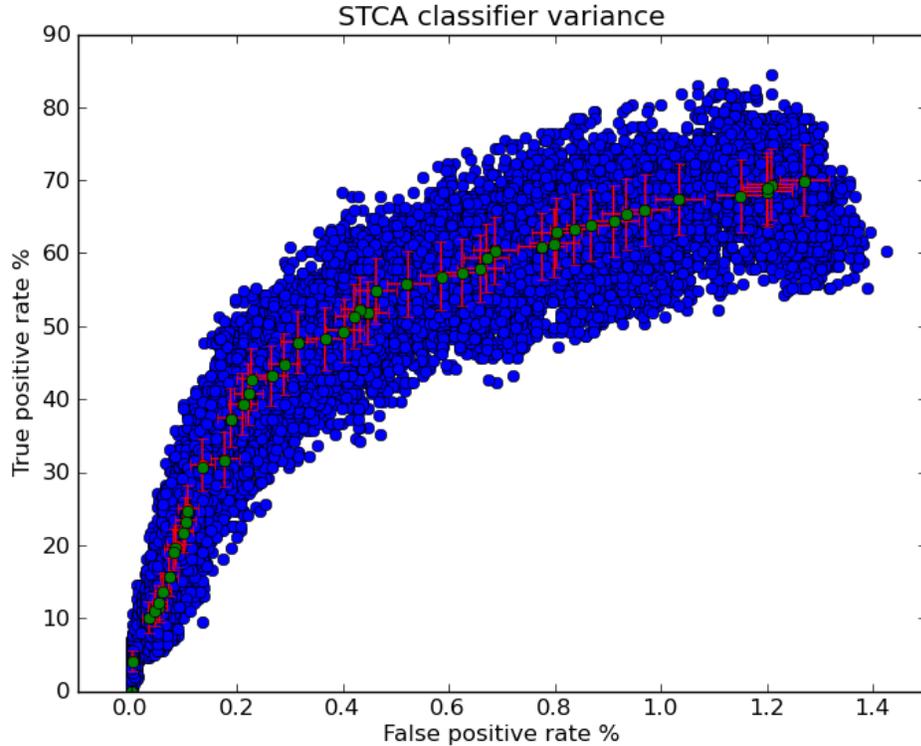


Figure 7.14: STCA variance estimated using the bootstrapping technique. Green circles represent the original STCA classifier positions (Multi-archive optimiser 11900 iterations, MACC 2 weeks corrected dataset), blue circles the re-evaluated TP/FP rates on 500 surrogate (bootstrapped) datasets. The error bar locations are centered at the means of the surrogate dataset, the bars themselves represent one standard deviation.

Probabilistic archive $\alpha = 0.7$, Probabilistic archive $\alpha = 0.8$ and Probabilistic archive $\alpha = 0.9$ all performed better, reducing under-fitting; Probabilistic $\alpha = 0.8$ being the closest to the optimal AUC at 0.809, -0.064 from the optimum.

Probabilistic archive with $\alpha = 0.8$ is by far the most effective method of reducing variance that we have observed. In the next section we discuss application of this technique to optimisation of STCA.

7.3 Reducing STCA classifier over-fitting

To underline the issue of over-fitting in STCA parametrisations, figure 7.14 shows the estimated variance of archive members ascertained via the bootstrap method of re-sampling dataset instances. As we can see, the effect of using a relatively small optimisation dataset (two weeks) on TP rate can be quite severe, undermining confidence in the consistency of the system. Note that the FP rate of parametrisations is relatively constant due to the severe class skew present in STCA datasets.

In section 7.2.7, the Probabilistic archive with $\alpha = 0.8$ was shown to be most successful at reducing over-fitting, therefore we applied this method of optimisation to STCA. Figure 7.15 shows preliminary results. In this trial the multi-archive optimiser was used, each regional

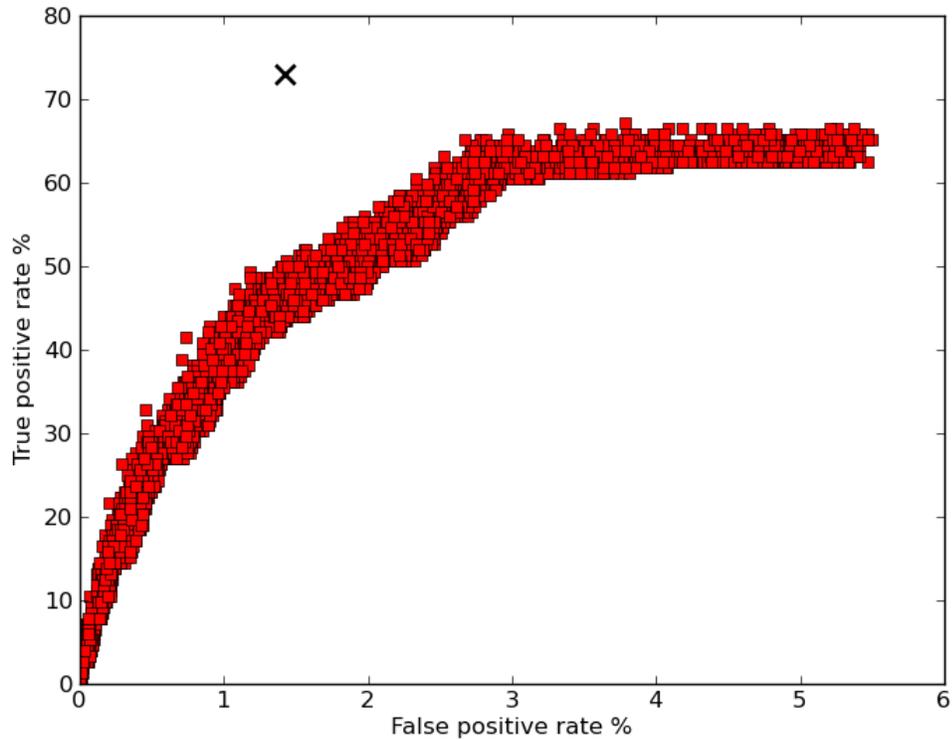


Figure 7.15: The result of optimising STCA using the multi-archive optimiser with a probabilistic archive on an MACC dataset (data covered a whole month and included odd days only, with corrected and uncorrected region assignments). Here the overall archive is shown after 11900 iterations, $\alpha = 0.8$.

archive employing the probabilistic comparison method in order to determine dominance. Disappointingly, the resulting archive appears to be under-trained, the NATS Operating Point at 73.03% TP, 1.42% FP highlighting the extent to which this is the case. We note that, although the width of the archive is relatively thin, it is very dense containing 3579 members, as even small variations in TP/FP rate are determined to be *possibly* non-dominating.

When applying this technique to STCA, several problems present themselves. As illustrated in figure 7.15 the size of the archive is substantially increased. Previously, archive size had been of little concern as the elitist nature of the optimiser appeared to naturally keep numbers down to a practical figure. With such a dense population, computational complexity and memory constraint issues become more significant, as the time taken to compare, sort, update etc. is multiplied. Fortunately, in the case of STCA the time taken to update a probabilistic archive (and complete other house-keeping tasks) is still insignificant compared to the time taken to evaluate parametrisations on the STCA system, however, this may not be the case in other problems domains. Memory constraints may also be problematic, depending how the optimiser is implemented with regard to memory footprint of parametrisations and real-time visualisation of the estimated Pareto front.

Perhaps more importantly, the substantial increase in archive size can affect the rate of

convergence. In explaining why the archive in figure 7.15 is so poorly converged, it is possible that the population increase has resulted in a severe drop in selective pressure. Permitting so many similar, potentially dominated, members to be selected effectively means that the whole archive must inch forward, as each member must be dragged in turn, impeding convergence. Unfortunately the dataset used for optimisation in figure 7.15 is different to that of previous trials so direct comparisons of convergence are not possible, however, comparison to the corresponding NATS OP is enough to ascertain that development of the Pareto front is stunted. It is also possible that the drop in selection pressure is exacerbated by the use of the multi-archive optimiser.

To reduce archive size a method of pruning parametrisations is needed. Possible suggestions are to implement some sort of niching method, or to include a region of influence around each currently archived parametrisation and not permit entrance of any new candidate that would encroach/intersect this boundary in objective space. Single-parent mode may also be of use if we were to accept ‘probably’ non-dominating parametrisations into the archive, but only move to dominating/mutually non-dominating points; retaining the diversity of probably non-dominating parametrisations in the archive, whilst maintaining selection pressure by using a non-probabilistic single-parent from which to derive child parametrisations. Alternatively, appropriately tuning α to the problem domain may be the simplest solution.

Another barrier in application of the probabilistic approach to STCA concerns the ‘validation’ phase (outlined in section 7.2.7), in which a second subset of data is used to select and generate an estimated Pareto front from all probabilistic members. Datasets currently available at NATS tend to extend to one month’s worth of data. Although many such datasets exist, it is not advisable to combine them as aircraft movements contained may correspond to trends or airspace structures that no longer exist. We could, of course, divide the month into subsets (as we have done for training/test sets). However, in order to effectively train, validate and test the probabilistic archive we require a minimum of three subsets. By dividing a month’s data in such a fashion we are likely to end up with datasets that contain a very coarse level of detail. Expecting accurate classification of trends on such sets may be optimistic.

Unfortunately due to time constraints these issues will not be further investigated in this work. A possibility for future work would be to run more trials using alternative α values and optimisation modes (e.g. single-archive optimisation). Alternative starting conditions such as using a converged Pareto front from one of the earlier runs could also be tested and techniques to reduce the probabilistic archive’s size whilst maintaining diversity could be explored.

Chapter 8

Conclusions

In this final chapter we suggest potential uses for an STCA ROC in visualisation and comparison of systems and airspaces. We summarise work presented in this thesis and outline future avenues of investigation.

8.1 Applications

Perhaps the principal use of the evolutionary algorithm is to optimise STCA performance for a particular airspace and, with the optimal ROC curve on hand, personnel may make a reasoned choice as to the false alert rate that must be tolerated to achieve a particular true positive alert rate, or vice versa. However, the optimal ROC curve also has a number of other potential applications which will be discussed in the following sections.

8.1.1 Comparison of STCA systems: Optimisation of ‘Enhanced STCA’

It is important to be able to assess the changes, such as software upgrades, made to STCA systems themselves. By generating the optimal ROC curves for a pair of systems (using the same training dataset) the performance of the two systems may easily be compared. Clearly, if one system’s ROC curve dominates another’s then the first is to be preferred.

As an illustration figure 8.1 overlays the ROC curves for STCA and ‘Enhanced STCA’ (ESTCA). ESTCA contains a number of modifications and enhancements to the previous Issue 4 specification of NATS’ STCA. Primarily ESTCA introduces a new fine filter to predict the lateral path of aircraft holding in stacks; and secondly it allows downlinked selected flight level information to be used to provide improved alerting performance. There are also smaller changes to existing STCA logic. As a result there 3863 optimisable parameters, in contrast to 1417 for STCA.

Due to time constraints the dataset used contained half the number of aircraft track pairs that would normally be used for manual optimisation; $\approx 73,000$ track pairs from London Terminal Control Centre (LTCC), July 2006. In order to increase the number of wanted alerts in the dataset, a fortnight’s worth of track pairs for which an alert is wanted were combined with a week’s pairs for which an alert should not be raised.

	STCA	ESTCA
Number of named parameters:	126	257
Number of optimisable named parameters:	89	224
Number of optimisable regions:	17	17
Total number of optimisable parameters:	1417	3863

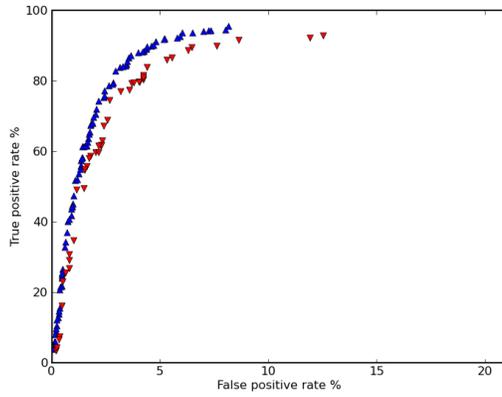
Table 8.1: Optimisable LTCC STCA/ESTCA parameters

Note how ESTCA takes roughly thirteen times as many iterations to obtain an ROC of similar quality to that output for STCA. A likely cause of this is the vast increase in parameter space (ESTCA has more than double the number of optimisable parameters - see table 8.1). Figure 8.1 thus serves to highlight the importance of ensuring the systems are properly tuned before comparison.

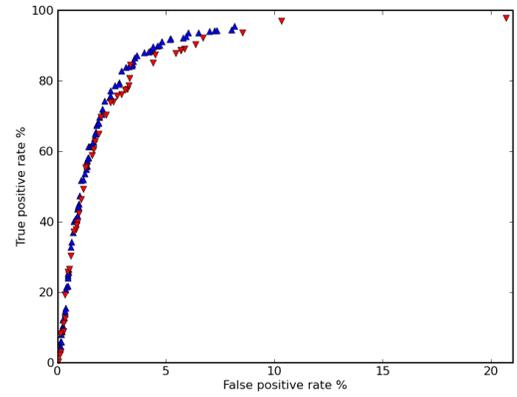
Figures 8.1 and 8.2 demonstrate that the optimisation software is capable of functioning on both a new airspace (LTCC) and an alternative STCA implementation (ESTCA), proving that it is capable of matching (and even exceeding) the manually tuned NATS operating points (marked as a black cross in 8.2). Figure 8.2 illustrates ROC progression after 2656 iterations, the same as figure 8.1(a), however the two systems' results have been separated for clarity and the associated COP marked.

After 34846 iterations (figure 8.1(c)) progress of the ESTCA front seems to have slowed to a stop. Assuming that we have obtained a near optimal ROC for the ESTCA system, why then does it not show any major improvements over the old STCA system? Firstly, the additional logic primarily concerns the behaviour of aircraft in stack of which there will be relatively few alerting instances compared to the dataset as a whole, thus we are unlikely to see their effect on the ROC in terms of a change in overall TP and FP rates. Secondly, many of the new parameters are a result of previously non-regional parameters having been split into multiple values, one for each region. This was done to increase the flexibility of the system, however, it means that to all intents and purposes the STCA logic and thus optimal configuration remain largely the same. What we *can* say about the new system is that it does not appear to perform any worse than the old.

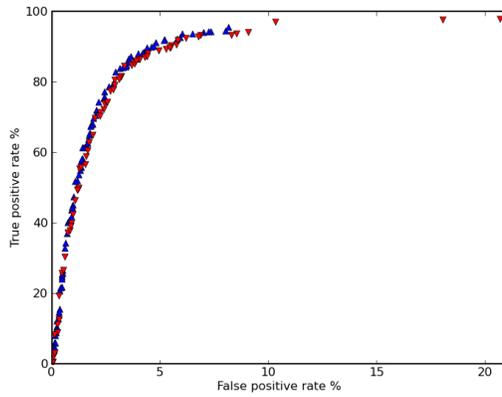
As Europe moves towards standardisation of STCA systems and their operation [32], an exciting possibility is that the optimal ROC curve may serve as a measurement tool for comparing STCA systems. It is well known from statistical pattern recognition that the area under the ROC curve is a measure of a classifier's ability to distinguish between two classes (here wanted and nuisance alerts) [36]. Although the total area under the Pareto ROC curve is likely to be less relevant than the area dominated by the region close to operational parametrisations, the optimal ROC curves of two STCA systems optimised on the same data provide a direct comparison of the performance of the two systems. Thus, there may be potential for developing a certification of STCA systems reaching a given standard. However, it is recognised that curves are only comparable for systems optimised on the same airspace using common data. Thus a lack of common data and varying STCA architectures is likely to be the greatest barrier to assessing systems in this way. Additionally, a reluctance of vendors



(a) STCA vs. ESTCA (2656 iterations).



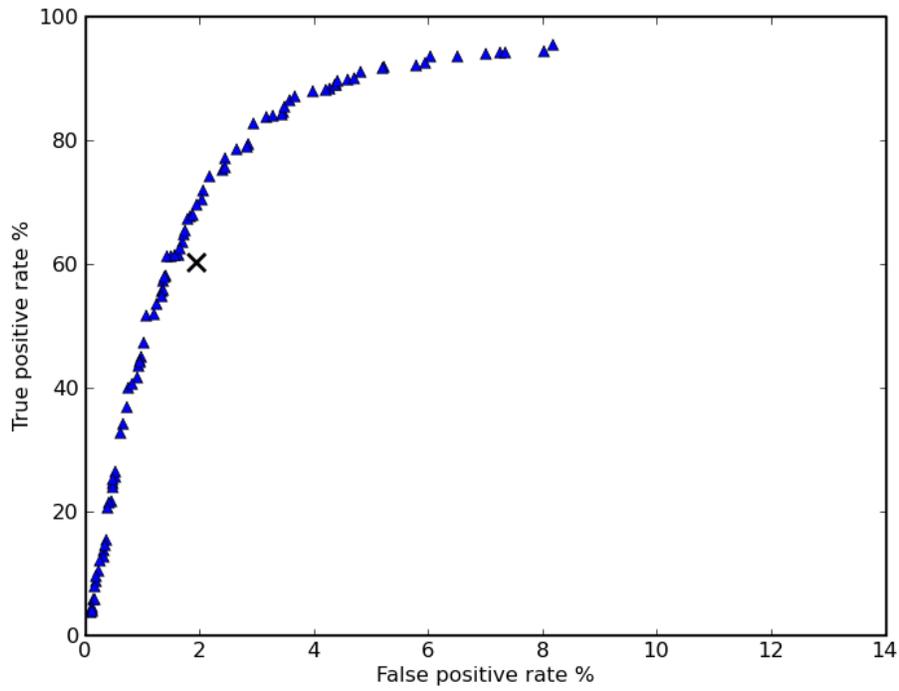
(b) STCA vs. ESTCA (2656 STCA iterations, 18009 ESTCA iterations).



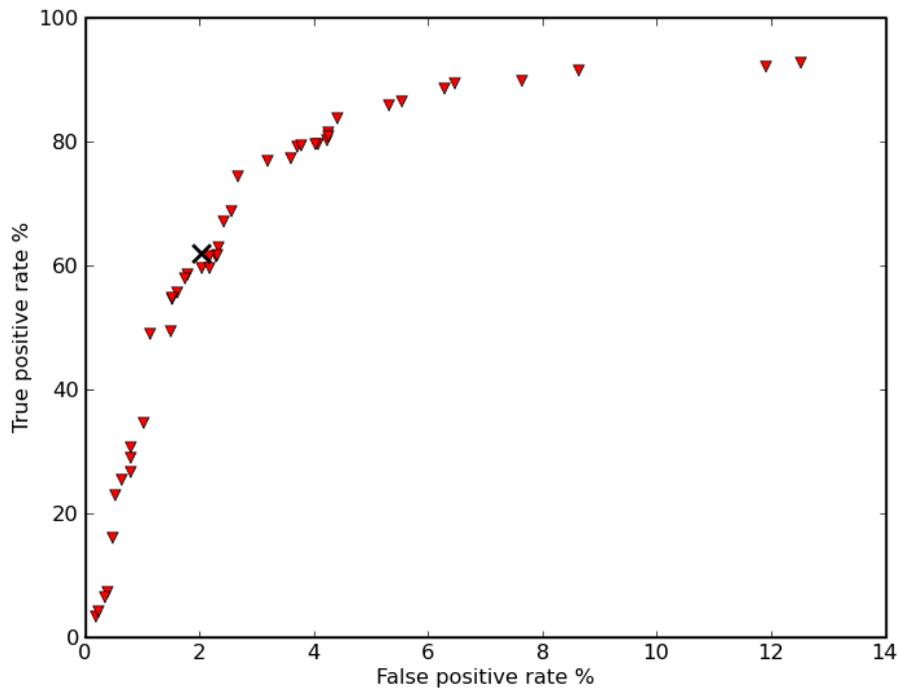
(c) STCA vs. ESTCA (2656 STCA iterations, 34846 ESTCA iterations).

Figure 8.1: Comparison of STCA vs. ESTCA, Multi-archive, aggressive optimisation results, LTCC dataset.

to compare systems could present a problem.



(a) *STCA* Multi-archive, aggressive optimisation, LTCC dataset (2656 iterations). NATS Operating Point: 60.31% TP, 1.93% FP.



(b) *ESTCA* Multi-archive, aggressive optimisation, LTCC dataset (2656 iterations). NATS Operating Point: 61.96% TP, 2.01% FP.

Figure 8.2: Multi-archive, aggressive optimisation results, LTCC dataset.

8.1.2 Visualisation

Arguably the primary use of an STCA ROC is as a decision making tool, illustrating the expected penalties associated to selecting a particular configuration. With the optimal ROC curve on hand, personnel may make a reasoned choice as to the false alert rate that must be tolerated to achieve a particular true positive alert rate, or vice versa.

The optimiser provides an additional tool on which to base decisions, previously the receiver operating characteristic of the STCA system was unknown. The range of possible options was harder to ascertain. Where before the manual optimisation process resulted in a cluster of parametrisations based around the previous operating point, now personnel can decide where best to be on the ROC curve and select the appropriate STCA configuration for that TP/FP rate. The ROC provides a readily interpretable graph of the best possible trade-offs between TP and FP rates, allowing operating parameters to be chosen with a full knowledge of the alternatives.

The ROC can also be used as a means of demonstrating to non-technical personnel the relationship between TP and FP in an instantly comprehensible manner. The ROC communicates why it is not possible to achieve a 100% TP rate, or rather that by doing so you would essentially render the system useless, as the disproportionately high number of resulting nuisance alerts would be inhibitive. It is bad for air traffic controllers to rely too heavily on the system, as STCA can miss potentially serious aircraft encounters. Conversely, if STCA were to continually alert on harmless situations, controllers would lose faith in it, assuming that every encounter was bogus; this too is undesirable. The ROC can help ascertain the correct level of trust to place in the STCA safety-net.

8.1.3 Airspace review

The ROC by nature represents an optimisation for a particular setup and associated air traffic pattern. You cannot use the optimisation process to trial potential changes to an airspace because the resulting radar data/traffic patterns will be different before and after the change. However, ROC comparison could provide a means to assess the changes' effect on the STCA system.

We have to be careful to note that ROC comparison between configurations does not provide a measure of how effective the new airspace is. It gives a comparison of what effect the airspace change has had in terms of what can now be achieved with the STCA system. It is a measure of STCA system performance.

As an example, referring to figure 8.3, if the green ROC curve marked an STCA optimisation before an airspace change and the red ROC the subsequent optimisation after that change. The drop in ROC TP and FP rates could indicate that something in the airspace change has adversely affected the STCA system, allowing it to be marked for early review.

In addition, the particular region of airspace that yields changes in performance may be identified, possibly leading to further reconfigurations. In fact the multi-archive optimisation mode automatically generates regional ROCs (see figure 8.4 for an example of multi-archive

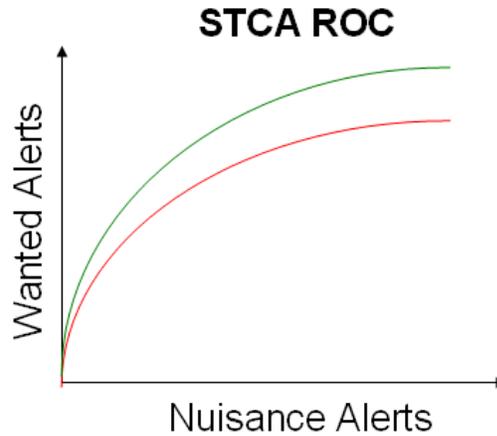


Figure 8.3: Fictional comparison of ROCs before and after airspace review.

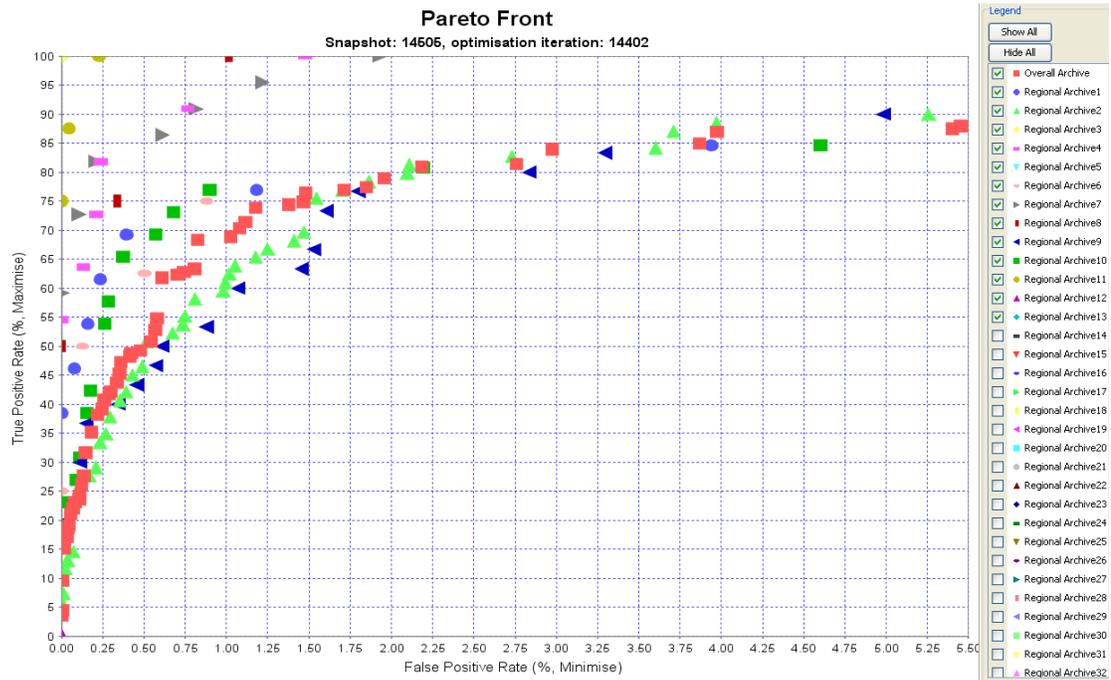
output with all regional archives displayed, the ‘overall’ front for the airspace is marked as red squares).

However, it should be noted that a better alert rate does not necessarily mean better airspace design in terms of an air traffic controller’s work load or difficulty in controlling the airspace. The optimiser simply provides an additional tool for analysis of airspace reconfiguration. It permits STCA maintenance personnel to assess changes in safety-net performance secure in the knowledge that any change observed is due to airspace reconfiguration rather than a sub-optimal configuration of the STCA system.

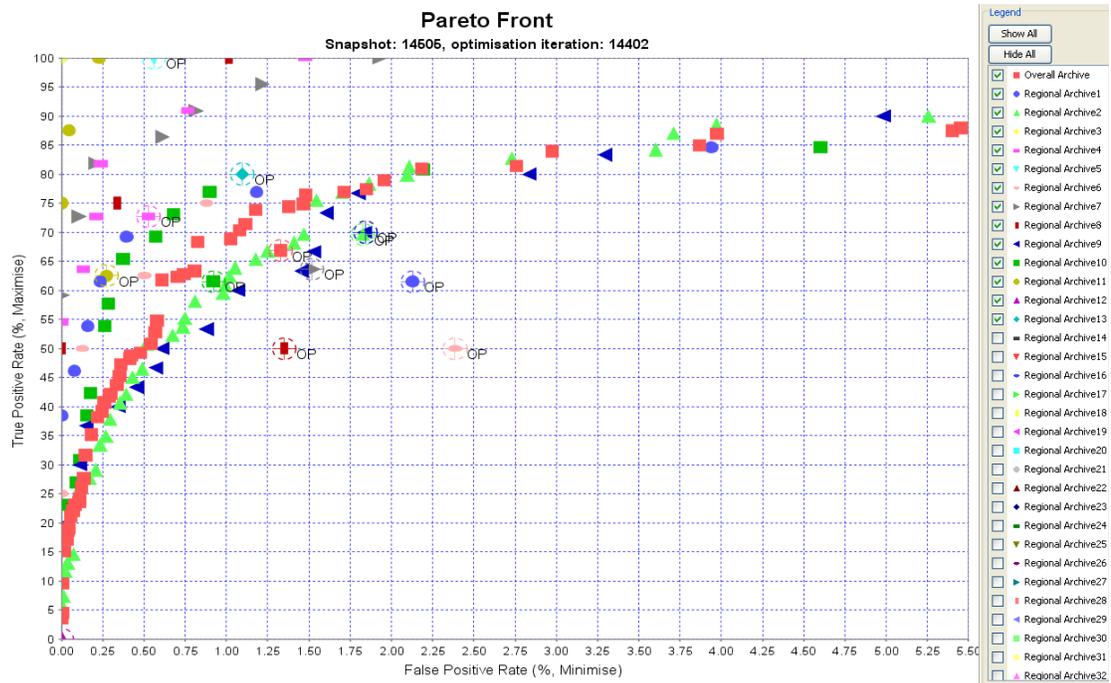
8.1.4 Summary

In addition to more obvious uses as a decision making and communication aid, this section has suggested how the optimiser might be used as a new tool for comparison of STCA systems and airspace review. A special note of the following should be made:

- Direct comparison of different STCA systems can *only* be made if the optimisation of them is performed on identical datasets.
- Assessment of an airspace changes’ effect on STCA system performance can be made, as long as it is noted that the underlying traffic patterns in the dataset will have been altered; we are monitoring what effect reconfiguration has in terms of TP and FP rates now achievable with the STCA system. The ROC is *not* a measure of how effective an airspace is.



(a) Output without NATS current operating point marked.



(b) Output with NATS current operating point included.

Figure 8.4: Illustration of regional fronts generated by the multi-archive optimiser.

8.2 Conclusions

We have described a straightforward evolutionary algorithm for the simultaneous optimisation of the trade-off between genuine and nuisance alerts for STCA systems. The methodology is applicable to any STCA system and airspace and is currently being put into use at NATS as part of the parameter review and optimisation process.

The major limiting factor of the automated optimisation process is the time taken to evaluate candidate parametrisations on the off-line STCA model. We have sought new methods of dividing the parameter space for perturbation, demonstrating that by treating STCA regions as a set of separate optimisations to be performed in parallel (chapter 4), we can obtain an overall ROC of equal or better progression in less than half the number of iterations of a ‘traditional’ single-archive approach (chapter 3).

We note, however, that the effectiveness of multiple regional archives is intrinsically linked to the independence of regional parameters and that dependencies may introduce noise to the candidate parametrisation. In the case of the ‘multi-archive’ mode, each new ‘child’ parametrisation is still evaluated with the STCA model, allowing the detrimental effects of noise to be minimised. On the NATS STCA system the multi-archive optimiser was very effective. Conversely however, the ‘estimated-archive’ optimiser was not so successful (chapter 4, section 4.2).

The estimated-archive optimiser highlights the necessity of independent regional parameters. This optimisation mode uses an estimate of parametrisation fitness to determine dominance, following a recombination of *evaluated* regional parameters with existing estimated-archive members. As optimisation progresses the true (evaluated) fitness of estimated parametrisations and the estimated fitness become increasingly disparate, eventually inhibiting further ROC growth. This is due to noise introduced to the estimated parametrisations. Although we have suggested possible methods of alleviating the effect of regional parameter dependencies, showing that a ‘corrected’ dataset can improve the ROC located by multi/estimated-archive modes (chapter 4, section 4.3), the presence of cross-regional aircraft encounters and non-regional parameter values are a continuing source of dependency between regional subsets.

In chapter 5 we described a method of ‘safely’ exploring unknown parameter space. This optimisation was dubbed *aggressive* because we did not constrain the parameters to lie within ranges previously used when manually tuning STCA. The additional freedom allowed to the parameters permitted an optimal ROC curve to be found that dominated the conservatively optimised curve, generating a broader ROC that reached out into higher TP and FP rates. Combining the aggressive perturbation methodology with that of the multi-archive optimiser generated the most optimal ROC curves seen for the NATS STCA system, in the smallest number of iterations.

In chapter 6 we investigated an alternative representation of the optimisation state; maintaining only a single parent throughout all generations, rather than selecting from an archive. Prior work [72, 73, 61] suggested that this may improve convergence rates, however

this was not found to be the case when applied to the problem of optimising STCA.

The effect of non-regional parameter dependencies was also investigated in section 6.4. We found that disabling optimisation of the non-regional parameters can indeed aid convergence. However, whether this is due to a reduction in noise, a reduction in search space complexity or simply the incorporation of, more optimal, NATS COP values is inconclusive.

In chapter 7, we took an initial look at the effects of overtraining on classifier variance. The STCA optimiser was found to quite severely overfit parametrisations to the training dataset. Several methods for reducing the variance seen between training and test dataset TP/FP rates were subsequently explored using a simplified neural net classification problem. Of the schemes applied, the ‘probabilistic dominance’ based technique was found to significantly outperform all others. However, we are unable to ascertain whether it could aid generalisation in STCA parametrisations due to inadequate datasets and limited time.

Finally, we have described potential uses for an STCA ROC, speculating how the optimal ROC might serve as a measurement tool for comparing STCA systems and airspaces.

Many industrial systems are structured in a similar way to STCA, we hope that techniques presented may be applicable to other highly parametrised problem domains.

8.3 Suggestions for future work

Here we mention a few suggestions for future work.

Investigate optimisation on a third objective

Although the optimiser is functionally capable (code is present) of performing optimisation on a third objective, in particular the ‘warning time’ given to wanted alerts, full investigation could not be carried out due to limitations with the current output from STCA. With modification of STCA to produce reliable warning time estimates, this would be straightforward. The current STCA model compares timings to a given base case (e.g. NATS Current Operating Point), to see which alerts gain or lose cycles – this tells us nothing about what the ‘ideal’ warning time for each alert *should* be.

As changes were made to the ‘Pap_Cam’ application¹, to permit regional alerts rather than merely the overall rate to be forwarded to the optimiser, this would seem the logical place to provide additional alert cycle information. However, this does require, relatively minor, additions to the datasets; a ‘desired warning cycles’ figure would have to be manually provided for each wanted alert instance. Provided we have this fixed point to calculate cycles lost/gained from, optimisation of the third objective is feasible.

Further trials using ‘probabilistic dominance’ to reduce over-fitting

In chapter 7 we took a preliminary look at various techniques designed to reduce over-training. Trials carried out on a toy neural network problem suggested that use of a probabilistic archive was most effective. Although an initial application to optimisation of STCA was presented, archive convergence was stunted. A possibility for future work would be to run more trials using alternative α values and optimisation modes (e.g. single-archive optimisation). Techniques to reduce the probabilistic archive’s size whilst maintaining diversity could also be explored. Slightly larger datasets will be required in order to train, validate *and* test on the STCA system itself.

Investigate composite classifiers

In the preceding chapters we have concentrated on improving the STCA system by seeking alternative, better, parameter configurations. Future work could look at how STCA might instead be improved by relatively minimal structural reconfiguration.

The substantial increase in computing power over the last decade now means that the STCA system itself may be considered relatively light-weight. As processing power increases it becomes feasible for much more complex logic to be integrated whilst upholding the requirement for real-time processing of aircraft tracks. Indeed, the NATS implementation of STCA has undergone many life-cycle reviews, the most recent incarnation being ESTCA

¹Pap_Cam: takes ‘raw’ output from the STCA system (usually segmented into days) and totals up alert information.

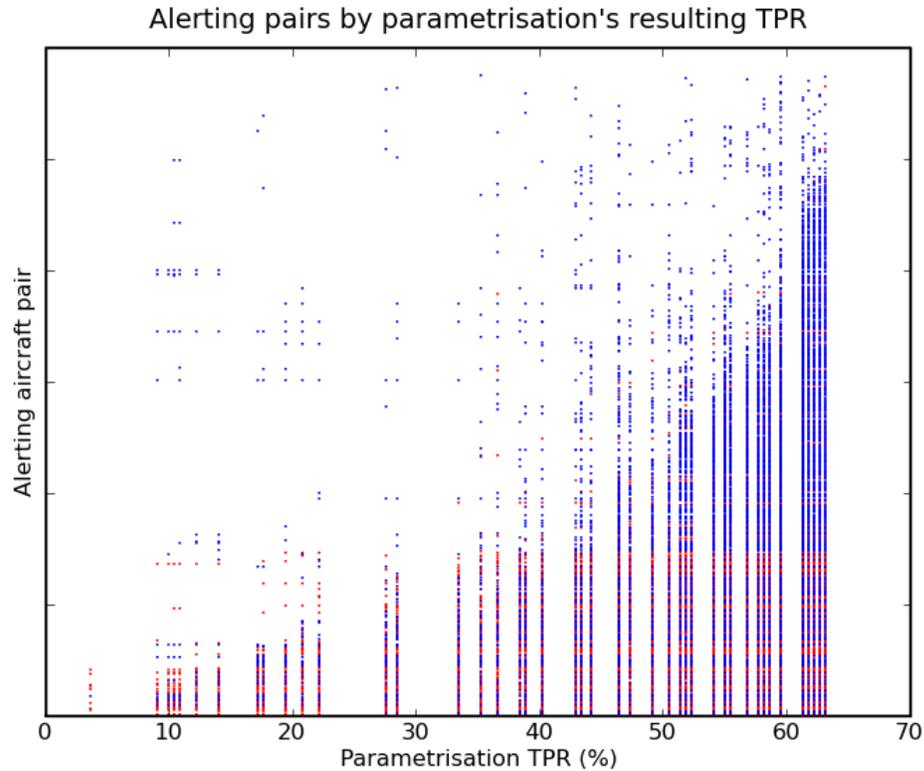


Figure 8.5: Alerting aircraft pairs vs. TP rate. Wanted alerts are shown as red points, unwanted as blue. Parametrisations taken from multi-archive optimisation (11900 iterations), of the MACC 2 weeks corrected dataset.

(see section 2.3.4, page 20). This permits us to consider techniques that might have been considered too costly or impractical a few years back.

A preliminary investigation into the effect of combining classifiers with different parameter settings in the hope of attaining TP/FP rates inaccessible to a single classifier was inconclusive, however this is a promising future avenue.

Figure 8.5 shows the constituent alerts generated from 48 parametrisations, taken from the overall archive of a multi-archive optimisation run after 11900 iterations (MACC 2 weeks corrected dataset). The resulting TP rate is shown, along with whether an alerting aircraft pair was wanted (red point) or unwanted (blue point). Aircraft pairs have been sorted according to how frequently a particular pair is found to alert throughout all parametrisations.

This diagram serves to illustrate whether an increase in TP rate is simply due to the addition of more, previously ignored, wanted pairs or, the loss of some existing wanted alerts and replacement by others (and vice versa for unwanted pair alerts): as we can see the latter is the case. From this we can establish that combining the output of several classifiers could potentially increase recognition of wanted alert pairs whilst inhibiting unwanted alerts because different parametrisations correctly classify different pairs. Also note how the diagram emphasises that the addition of a few extra wanted alerts, results in a comparatively vast number of unwanted alerts.

We suggest two possible methods of combining classifiers; random combination of classifiers, thus creating new classifiers that lie on the convex hull that contains the original constituent classifiers in objective space, and aggregation of multiple classifier outputs. In the former method, a classifier is selected at random (with some bias) to determine the outcome of processing each aircraft pair, whilst the latter method can be thought of as permitting multiple classifiers to vote; all classifiers contribute to the outcome of processing *every* aircraft pair.

It will also be useful to explore the effect on generalisation error when combining parametrisations that have been optimised using separate training and validation data subsets.

Appendices

Appendix A

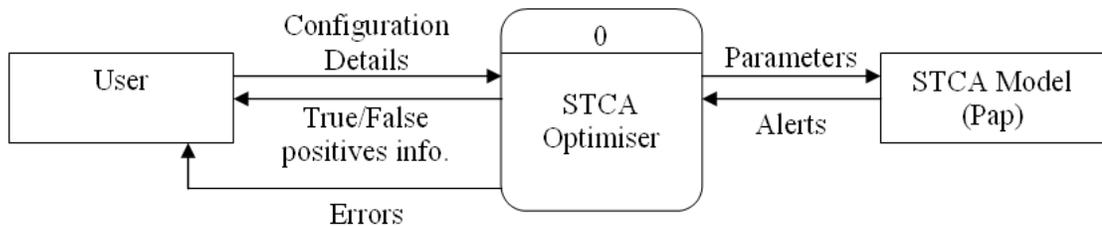
A.1 Example STCA parameter configuration file

The following shows an extract from an STCA parameter configuration file. Note that the full listing contains 3133 parameter values (STCA), 7015 parameter values (ESTCA).

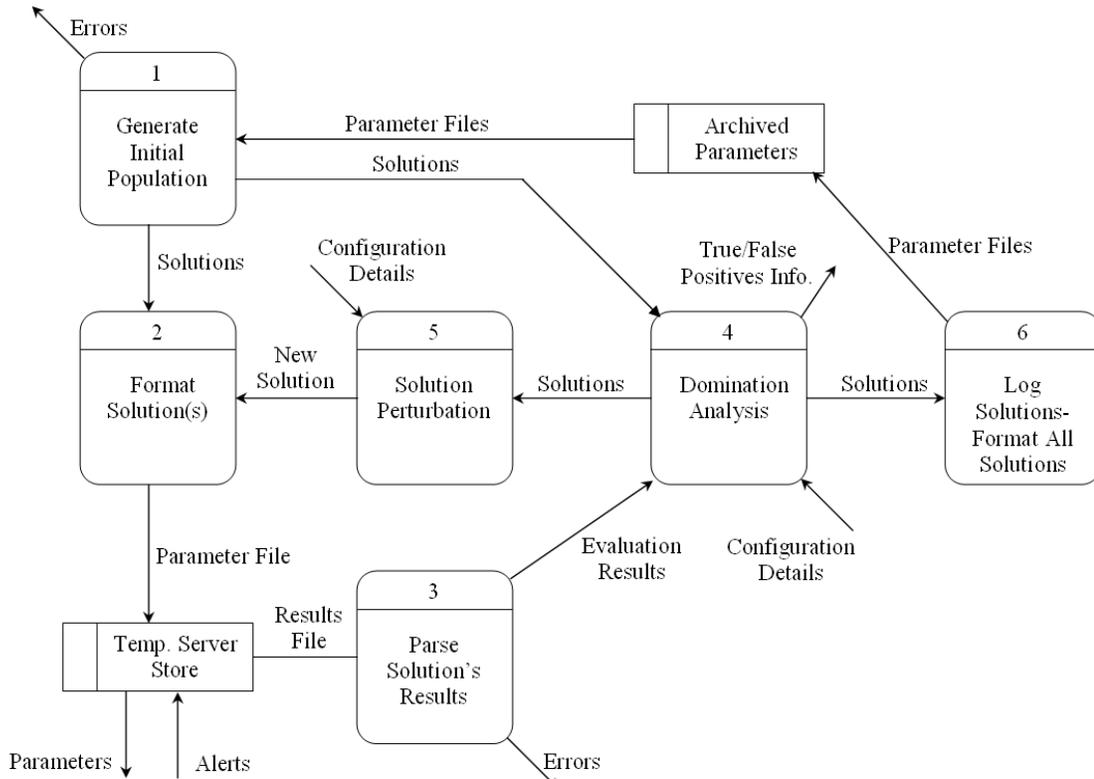
```
(General parameters)
GROUNDELEVATION    500 (ft)
HIGHLEVEL          29201 (ft)
CYCLE              4 (cycles)
...
(Fine filter region dependant values)
FFDIVFAST          120  120  120  120  120  ... (knots)
FFDIVSEPL          1.0  1.0  1.0  1.0  1.0  ... (NM)
...
(Linear prediction filter)
CPCORT             12   12   12   12   12   ...
CPFIRM             8    8    8    8    8    ...
LPLDA              0.75 0.75 0.5  0.75 0.5  ...
LPVDA              0.75 0.75 0.75 0.75 0.75 ...
...
(Current proximity values)
CPCORT             12   12   12   12   12   ...
CPFIRM             8    8    8    8    8    ...
...
(Manoeuvr hazard filter)
CPCORT             12   12   12   12   12   ...
CPFIRM             8    8    8    8    8    ...
...
(LP sliding window criteria)
LPWINDOW           3    4    4    2    3    ...
...
(CP sliding window criteria)
LPWINDOW           2    2    3    2    2    ...
...
(MH sliding window criteria)
LPWINDOW           4    2    3    2    4    ...
```

Appendix B

B.1 Optimiser data flow diagram



(a) Context diagram.



(b) DFD.

Figure B.1: Optimiser data flow diagram.

Appendix C

The following details the STCA application architecture and the method used to wrap it. The wrapper allows the optimiser to automatically invoke the STCA system and remotely oversee I/O.

C.1 Wrapping the STCA system

C.1.1 STCA architecture

- Pap4 (ardat_exe:pap_cam4 iss4): is a ‘.com’ file. These are script files containing OpenVMS DCL commands - they are not ‘true’ executables. Pap4 takes a series of user inputs such as desired radar, parameter and region files, and paths to wanted output files. It then generates a batch file (Pap_batch.com) designed to call ‘campap_cam4.exe’ repeatedly. ‘pap_cam4.exe’ then combines the results (e.g. encounter pairs over a two week period etc).
- Pap_batch.com: Is also a DCL command script. It contains repeated calls to Campap_cam4.exe and can be submitted to a queue of batch processes under OpenVMS.
- Campap_cam4.exe: is the actual STCA model responsible for generating the statistical results of alerts. It processes encounter pairs only.
- Pap_cam4.exe: is responsible for taking the ‘raw’ STCA output and combining the alert results from each day into a table of overall statistics.

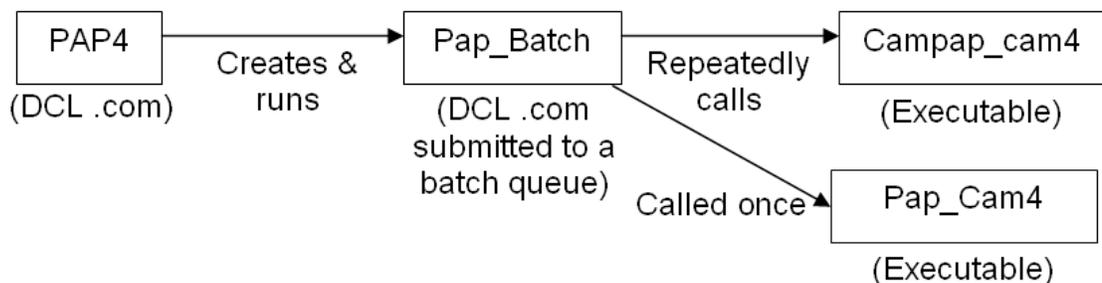


Figure C.1: STCA process diagram.

Executing a new STCA parameter evaluation

- 1). A modified version of Pap4.com is used to output a ‘template’ for use by the STCA Server. The output is almost identical to that produced by unmodified Pap4 - the only difference is that paths to parameter (.dat) and results (.Pap, .Wal) files have been replaced by ‘tags’.
- 2). The template is loaded by the ‘STCA_Wrapper’ class of the STCA server at instantiation.
- 3). Using string searches for the embedded ‘tags’ in the template, the wrapper writes out a version of the template with ‘desired parameters file to be evaluated’ and ‘results file location’ inserted.
- 4). Due to restrictions on the length of a command line passed by Java Runtime.exec() to OpenVMS, necessary arguments are output by the STCA optimiser Server to a separate file (‘Submit_Batch_Args.dat’). These are then read in by ‘Submit_Batch.com’ and used in adding an STCA batch run to the OpenVMS queue.
- 5). The wrapper calls the external .com file ‘Submit_batch.com’.
- 6). ‘Submit_Batch.com’ reads in arguments from the file ‘Submit_Batch_Args.dat’. These specify the location of the ‘Pap_Batch.com’ file to be executed, where the log from this process should be output to, and the unique name to be given to this process.
Using the information ‘Submit_Batch.com’ submits ‘Pap_batch.com’ to OpenVMS for batch processing.
NB: ‘Submit_Batch.com’ will not exit until the batch STCA evaluation process has completed. This is necessary as the STCA optimiser Server monitors the running status of ‘Submit_Batch.com’ in order to ascertain when evaluation terminates.
- 7a). ‘Pap_Batch.com’ contains repeated DCL calls to the ‘Campap_cam4’ (STCA) executable. STCA outputs alert details for each day of submitted data.
- 7b). ‘Pap_Batch.com’ calls ‘Pap_Cam4Mod’. This accumulates the result files produced by STCA and outputs the alert stats in a table.
NB: ‘Pap_cam4Mod’ is an alternate version of the ‘Pap_Cam4’ executable. It is a Java application developed in response to additional regional alert information being required by the optimiser.
- 8). When ‘Pap_Batch.com’ terminates a log is written out by OpenVMS.
- 9). The ‘Pap_Batch.com’ log is read in by the STCA optimiser Server. String searching is used to ascertain whether ‘Pap_Batch.com’ exited normally, crashed, or was aborted by a user.

The results of the STCA parameter evaluation may now be read in.

Terminating an STCA parameter evaluation

- K1).** The STCA Server wrapper calls 'Show_Entry.com', which executes the DCL command to show all user batch processes listed by entry Id number and name. The output from 'Show_Entry.com' is piped back to the wrapper.
Using string searching the 'Pap_Batch.com' process entry Id is obtained by looking up the unique process name that was given to it.
- K2).** The STCA Server wrapper calls 'Kill_Batch.com', passing the 'Pap_Batch.com' entry Id as a parameter at command line. 'Kill_Batch.com' executes the OpenVMS DCL command to terminate the 'Pap_Batch.com' process.

Appendix D

D.1 ARDAT CPU Usage Summary

ARDAT weekly usage patterns have to date been largely unknown. The STCA Optimiser in development is likely to process data $\approx 10,000$ times. A dataset of $\approx 140,000$ aircraft pairs can take 6 or 7 minutes to process per optimiser iteration (see figure D.5(a)). Thus, it is likely that the optimiser will have a significant impact on computational resources. In order to assess the optimiser's likely impact on departmental members, an analysis of busy/off-peak periods was performed over a 10-week period.

D.1.1 Optimiser configuration

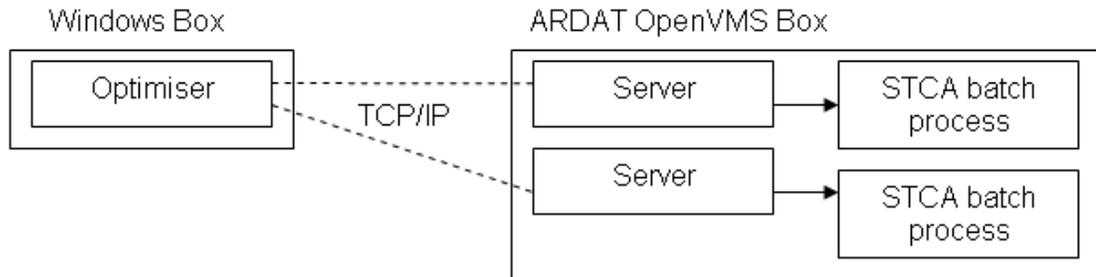


Figure D.1: Optimiser client-server architecture.

The core optimiser application is run from an entirely separate machine to the Alpha box and thus should have no impact on other ARDAT users. The Server application, used to wrap STCA, is not a CPU intensive process; it is simply used as a means to control execution of STCA batch processes remotely.

D.1.2 Monitor setup

A DCL script monitors the average CPU load over 15 minute periods with a sampling interval of 6 seconds. Results shown indicate how busy *all three* of the ARDAT CPUs are as an overall percentage.

The weekly boundary runs from Friday at 17:30.

D.1.3 Results

The following pages show graphs of average CPU usage recorded by week. The histograms indicate the number of times CPU usage reached a given level in that week (bins of 5%).

- Average weekly CPU usage is in the region of 30% (Figures D.2, D.3).
- The average weekly CPU usage during the hours of 9am to 5pm is around 40% (Figures D.2, D.3).
- In a 10-week period (70 days) the CPU load exceeded 40% on 28 days (Figure D.4(b)).
- The busiest days of the week for ARDAT usage are Monday and Tuesday, a drop in usage is seen Wednesday to Friday (Figure D.4(b)). System maintenance procedures (e.g. disk defragmentation) automatically run throughout the weekend, resulting in increased CPU activity in the early hours of Saturday and a gradual drop off towards Sunday (Figures D.2, D.3).
- In some minority cases the monitor does show that CPU usage reaches 100% (Figure D.4(a)). NB: a major contributor to this is the copying of many small files between a Windows box and ARDAT. This causes an ARDAT file server system service to ‘take-over’ available CPU. Unfortunately, limiting this processes priority is difficult due to it being an OS component. However, the need to copy files in such a fashion is a relatively unusual occurrence.
- The STCA optimiser server uses $< 1.5\%$ of one CPU’s resources.
- An STCA batch process will use 95-100% of one CPU’s resources (roughly 33% of ARDAT’s available CPU resource) however, it is set to a low priority allowing other tasks to take precedence.

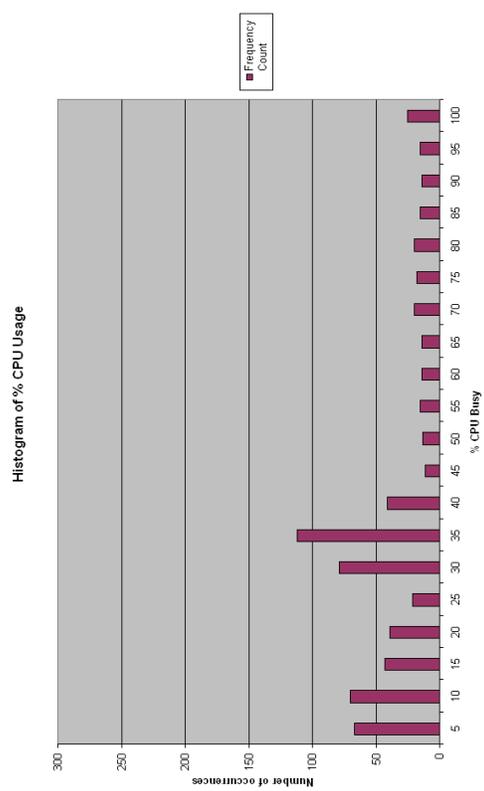
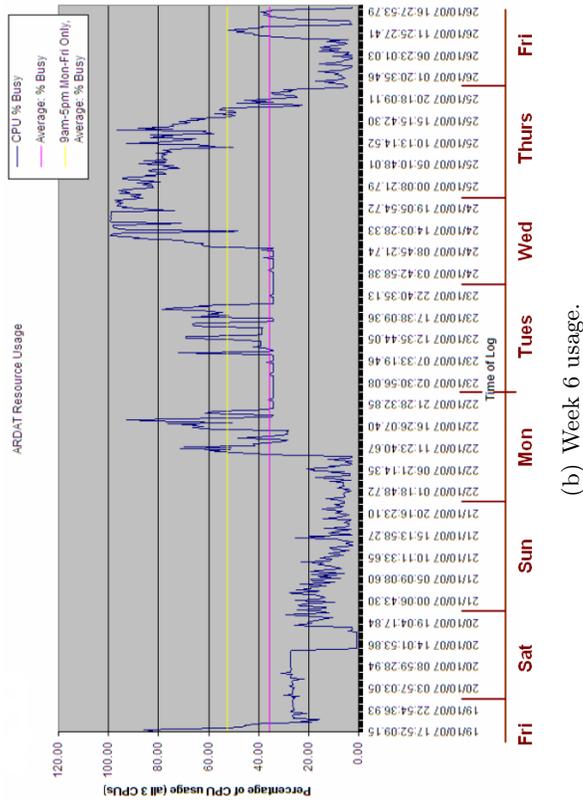
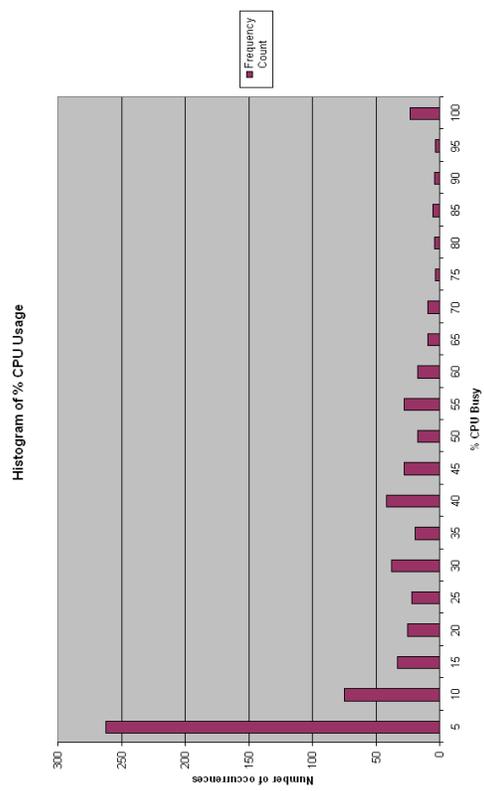
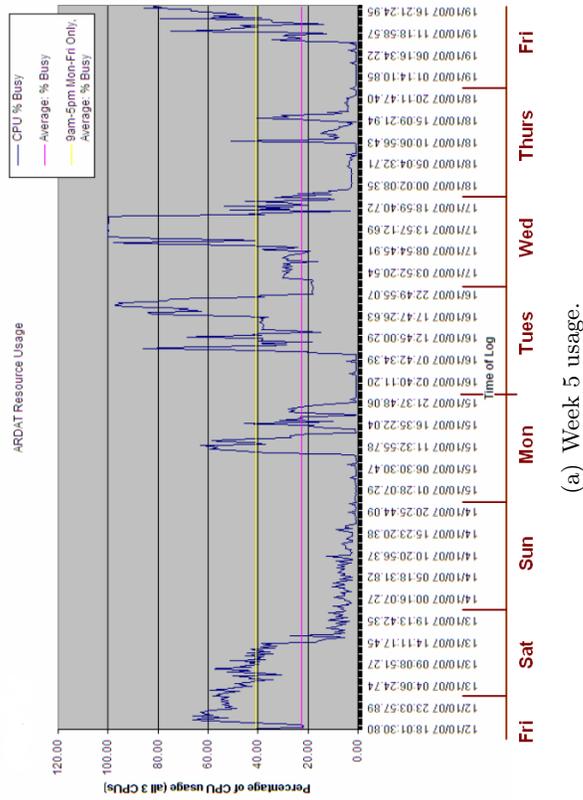
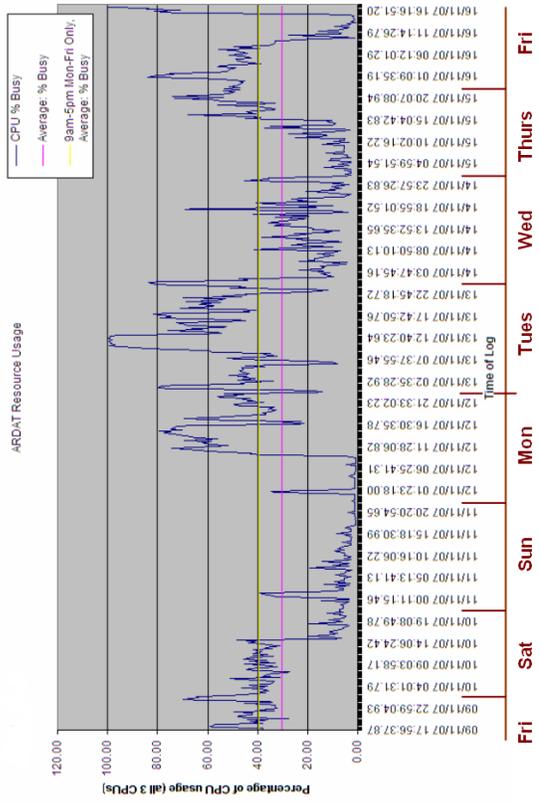
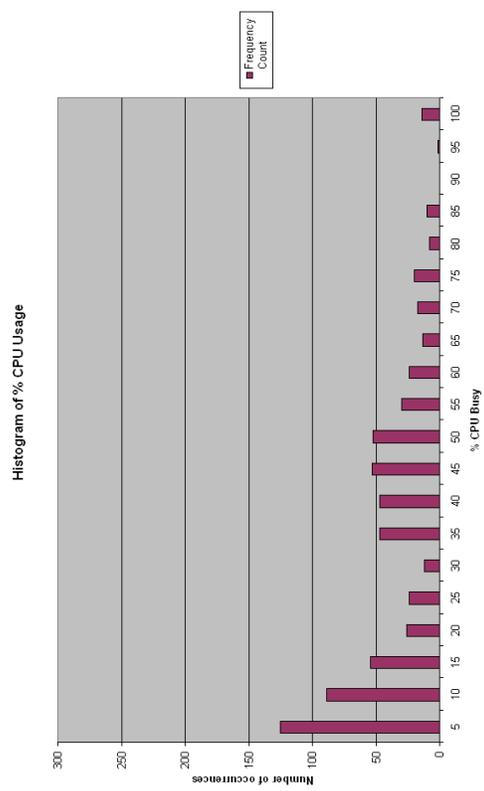


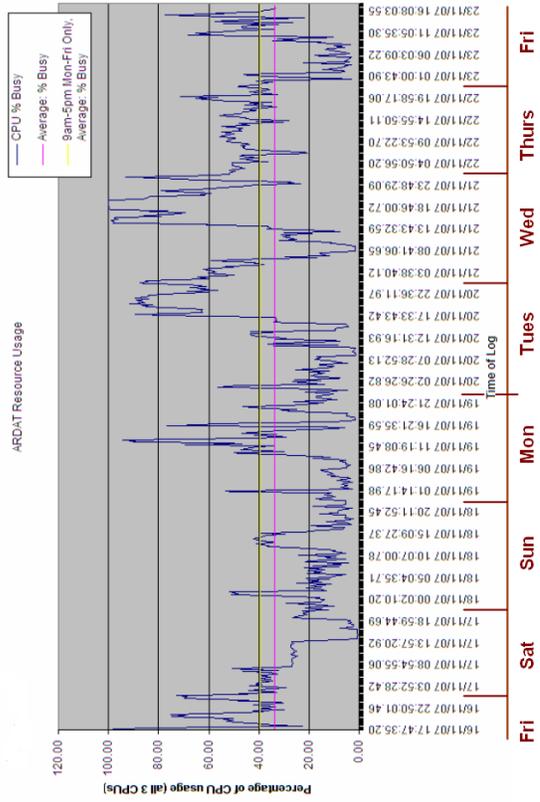
Figure D.2: Overview of weekly pattern: Weeks 5 & 6 (12/10/07-26/10/07).



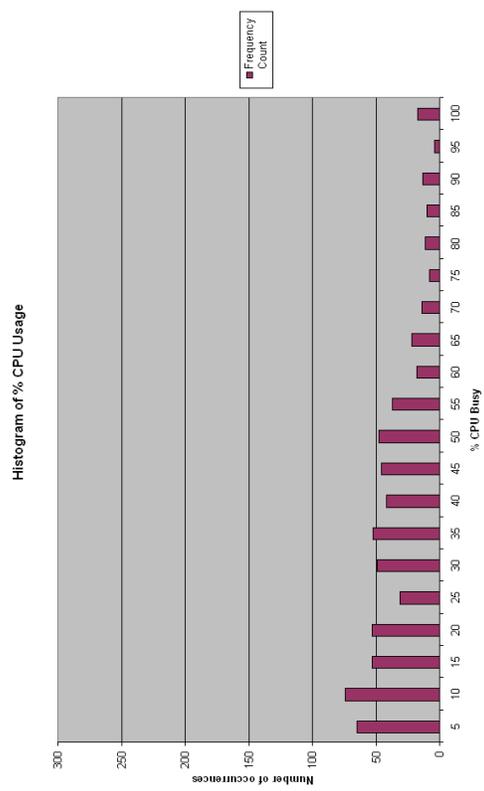
(a) Week 9 usage.



(c) Week 9 histogram of % CPU usage.

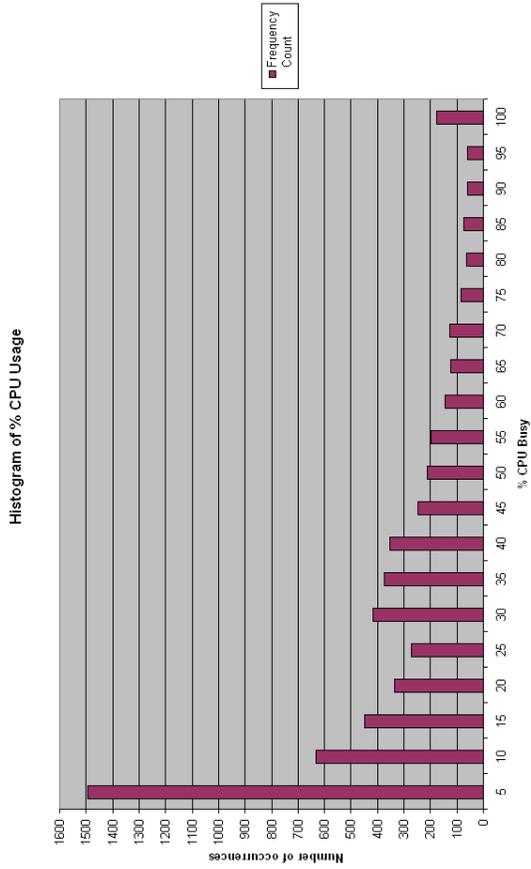


(b) Week 10 usage.



(d) Week 10 histogram of % CPU usage.

Figure D.3: Overview of weekly pattern: Weeks 9 & 10 (09/11/07-23/11/07).

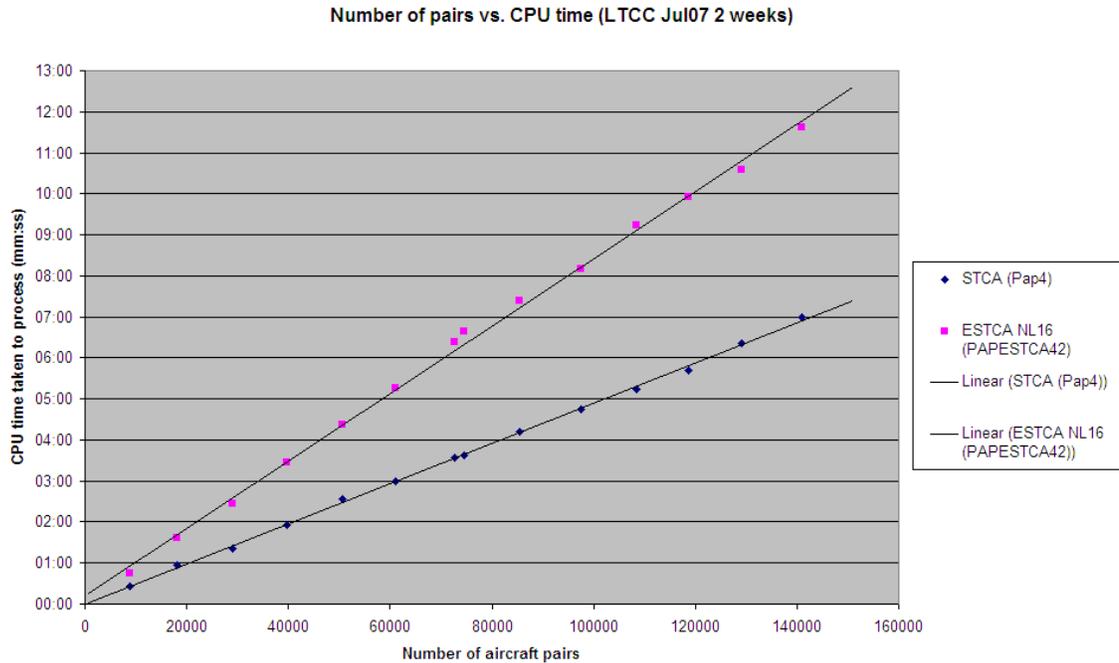


(a) Histogram of 10-week period (14/09/07 - 23/11/07).

Day of Week	Days where average CPU usage for the period 9am-5pm is above 40%										Summary	
	Week1	Week2	Week3	Week4	Week5	Week6	Week7	Week8	Week9	Week10		
Sat	0	0	0	1	0	0	0	0	0	0	0	Quiet
Sun	0	0	0	0	0	0	0	0	0	0	0	Quiet
Mon	1	1	1	1	0	1	1	0	1	1	1	Busy
Tues	1	0	0	1	0	1	1	1	1	1	0	Busy
Wed	0	1	1	0	1	1	0	0	0	1	1	Even Split
Thurs	0	1	1	0	0	1	0	0	0	1	1	Quiet
Fri	0	1	0	0	1	0	0	1	0	1	1	Quiet

(b) Incidents of CPU reaching above 40% during working hours (9am-5pm).

Figure D.4: Overview of 10-week period.



(a) Graph illustrating how the number of aircraft pairs in a dataset affects the resulting processing time of the offline STCA model.

Files used
STCA file: LAN_DISK:[OPT_DATA.PAPIN]NODEL_JUL07_RACT_NOCFL.PAPIN Parameters_V027.dat Regions_V038.dat
ESTCA file: LAN_DISK:[OPT_DATA.PAPIN]NODEL_JUL07_RACT.PAPIN STCA_PARAMS_NL_V031.dat STCA_REGIONS_NL_V043.dat ← Edited so that the bottom “DND London FIR” region was marked as ‘INACTIVE’ to make regions more consistent with original pap4 stca model/regions (better comparison) STCA_STACKS_NL_V002.DAT

(b) Files used for comparison.

Figure D.5: STCA & ESTCA, Number of aircraft pairs vs. CPU time.

D.1.4 Summary

It is clear from the results that despite the presence of some ‘busy’ periods where the ARDAT system becomes overloaded, for the moment at least, there are adequate resources to run the STCA optimiser without impinging on everyday activity. It should be possible to run one server near continuously; other servers can be scheduled to become active at ‘off-peak’ times e.g. weekends and evenings. The ability to scale optimisation runs should permit a good compromise.

In addition, the ARDAT system is scheduled for an upgrade in the near future as part of an LTIP project. As additional resources are not strictly necessary for the time being, no further action is to be taken regarding the KTP’s involvement in obtaining additional equipment.

Appendix E

E.1 Stills from optimiser mode comparison video

Each frame has the following layout-

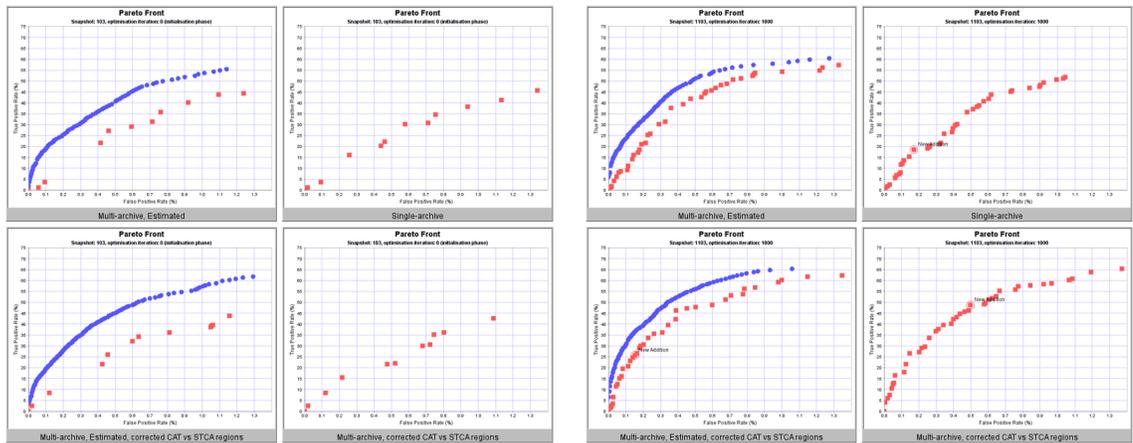
Top-left: Estimated-archive mode (estimated-archive in blue, ‘overall-archive’ in red).

Top-right: Single-archive mode.

Bottom-left: Estimated-archive mode, corrected dataset region labellings.

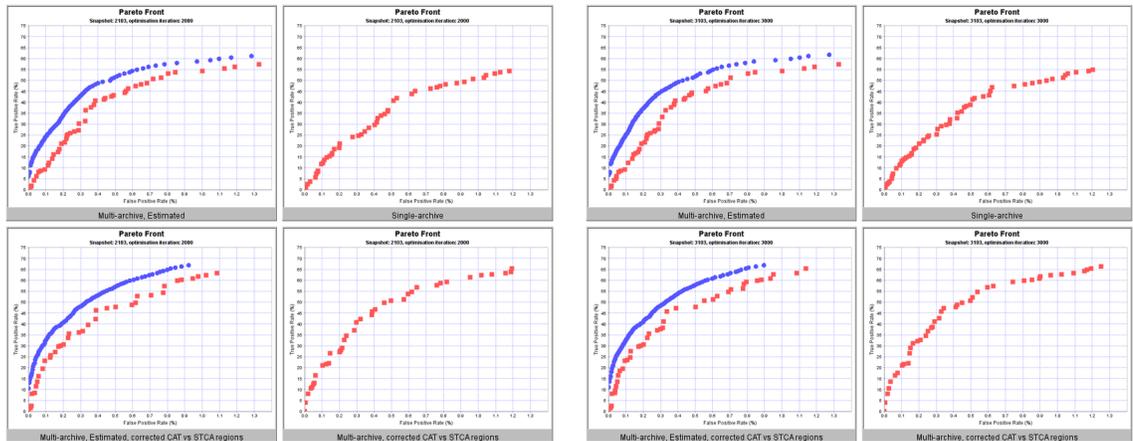
Bottom-right: Multi-archive mode, corrected dataset region labellings.

The dataset used was two weeks of MACC data (July 2007).



(a) Initial set.

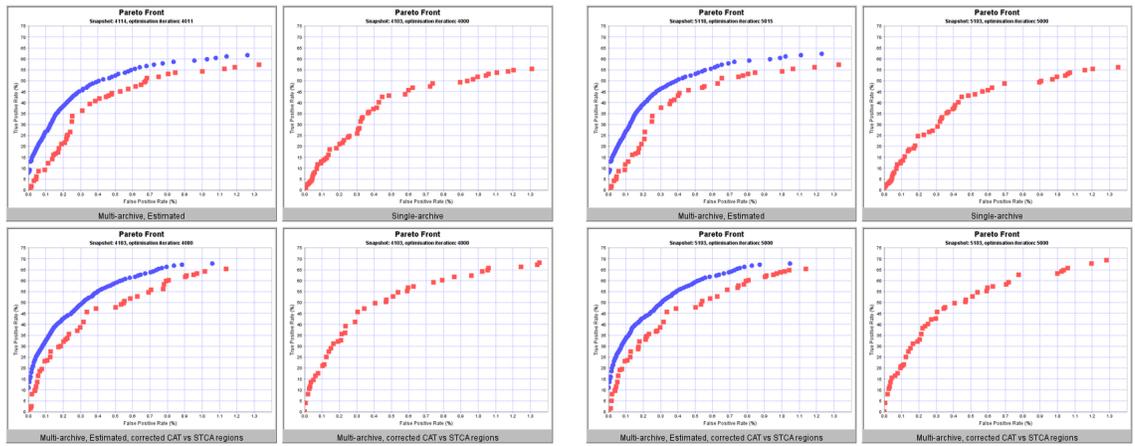
(b) 1000 iterations.



(c) 2000 iterations.

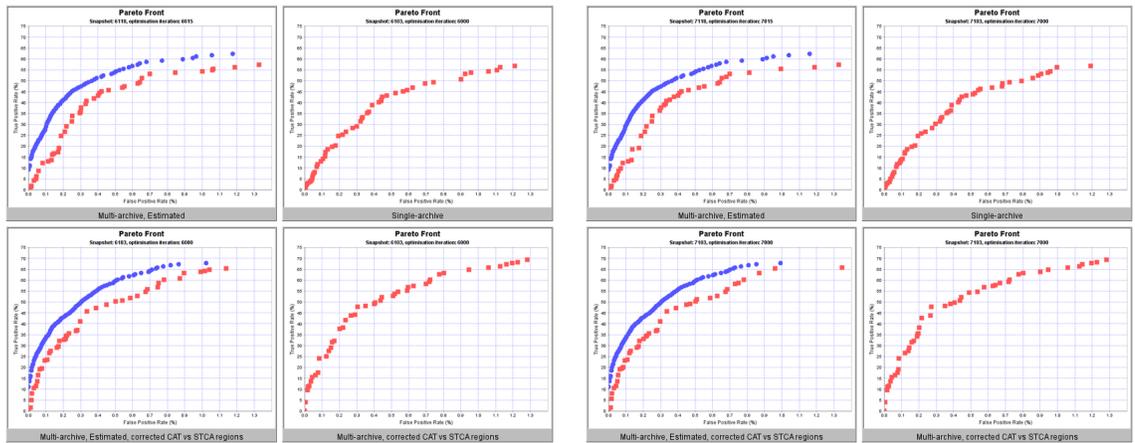
(d) 3000 iterations.

Figure E.1: Comparison video frames.



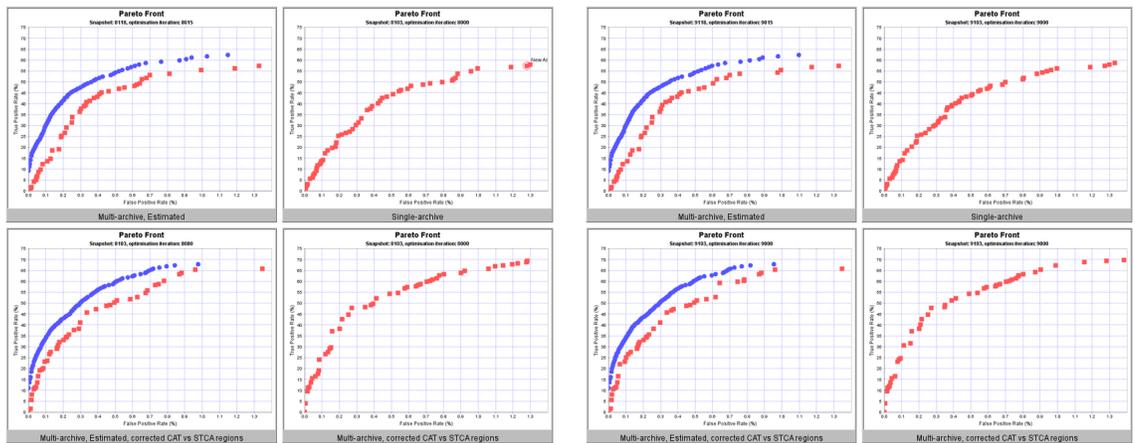
(e) 4000 iterations.

(f) 5000 iterations.



(g) 6000 iterations.

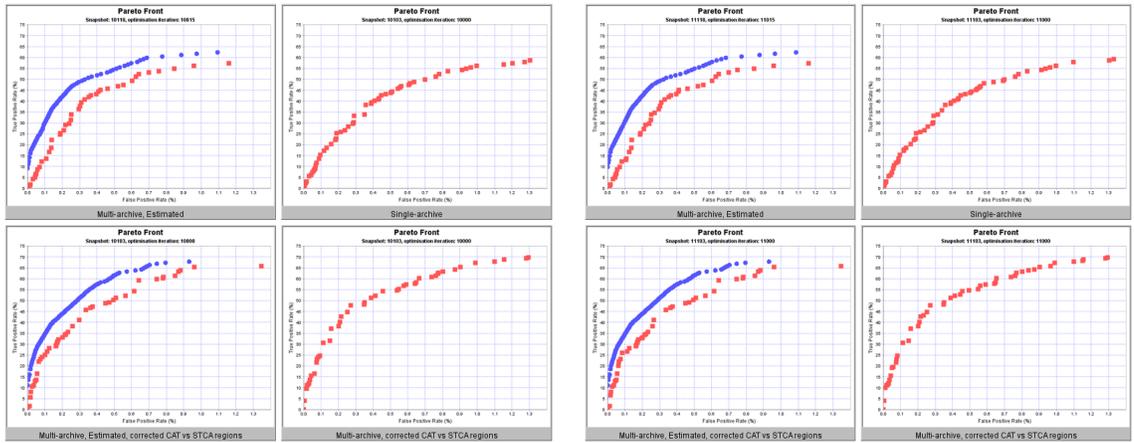
(h) 7000 iterations.



(i) 8000 iterations.

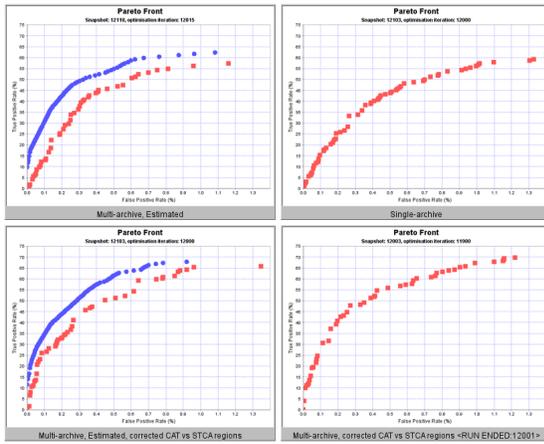
(j) 9000 iterations.

Figure E.1: Comparison video frames (continued).



(k) 10000 iterations.

(l) 11000 iterations.



(m) ≈ 12000 iterations.

Figure E.1: Comparison video frames (continued).

References

- [1] *Software Description for CAMPAP (Issue 1.03) Module*. NATS Ltd., Operational Analysis Department, 1st edition, August 2000.
- [2] P.J. Angeline. Adaptive and Self-Adaptive Evolutionary Computations. In *Computational Intelligence: A Dynamic Systems Perspective*, pages 152–163. IEEE Press, 1995.
- [3] P.J. Angeline. Two Self-Adaptive Crossover Operators for Genetic Programming. In P. Angeline and K. Kinnear, editors, *Advances in Genetic Programming II*. MIT Press, 1996.
- [4] T. Bäck. Self-Adaptation in Genetic Algorithms. In *Proceedings of the First European Conference on Artificial Life*, pages 263–271. MIT Press, 1992.
- [5] J.F.M. Barthelemy and R.T. Haftka. Recent Advances in Approximation Concepts for Optimum Structural Design. In *Optimization of Large Structural Systems. Proceedings of the NATO/DFG Advanced Study Institute*, pages 235–256. Kluwer Academic Publishers, 1991.
- [6] J. Beasley, H. Howells, and J. Sonander. Improving short-term conflict alert via tabu search. *Journal of the Operational Research Society*, 53:593–602, 2002.
- [7] G. Beeston, H. Howells, and M. Richards. *PAP User Guide*. NATS Ltd., Operational Analysis Department, 1st edition, August 2000.
- [8] R.E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957. Republished 2003: Dover Publications.
- [9] A. Ben-Tal. Characterization of Pareto and lexicographical optimal solutions. In *Multiple Criteria Decision Making: Theory and Application, Lecture Notes in Economics and Mathematical Systems (G. Fandel and T. Gal)*, volume 17, pages 1–11, 1980.
- [10] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [11] D. Büche, S. Müller, and P. Koumoutsakos. Self-Adaptation for Multi-objective Evolutionary Algorithms. In *Proceedings of the 2nd International Conference on Evolutionary Multi-criterion Optimization (EMO'03)*, pages 267–281. Springer-Verlag, 2003.

- [12] E. Cantú-Paz. A Survey of Parallel Genetic Algorithms. *Calculateurs Parallèles, Réseaux et Systems Repartis*, pages 141–171, 1998.
- [13] E. Cantú-Paz and D.E. Goldberg. Efficient Parallel Genetic Algorithms: Theory and Practice. In *Computer Methods in Applied Mechanics and Engineering*, pages 221–238, 2000.
- [14] C.A. Coello Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems. An International Journal*, 1(3):269–308, 1999.
- [15] C.A. Coello Coello. Evolutionary Multi-Objective Optimization: A Historical View of the Field. *IEEE Computational Intelligence Magazine*, 1(1):28–36, February 2006.
- [16] C.A. Coello Coello, A.D. Christiansen, and A. Hernández Aguirre. Using a New GA-Based Multiobjective Technique for the Design of Robot Arms. *Robotica*, 16:401–414, 1998.
- [17] C.A. Coello Coello and G.B. Lamont, editors. *Applications of Multi-Objective Evolutionary Algorithms*. World Scientific, Singapore, 2004.
- [18] C.A. Coello Coello and G.T. Pulido. A Micro-Genetic Algorithm for Multiobjective Optimization. In *Lecture Notes in Computer Science, Evolutionary Multi-Criterion Optimization*, pages 126–140. Springer, 2001.
- [19] C.A. Coello Coello and M.R. Sierra. A Study of the Parallelization of a Coevolutionary Multi-objective Evolutionary Algorithm. In *MICAI 2004: Advances in Artificial Intelligence*, Lecture Notes in Computer Science, pages 688–697. Springer, 2004.
- [20] G. Cybenko. Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2:303–314, 1989.
- [21] I. Das and J.E. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14(1):63–69, 1997.
- [22] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, 2001.
- [23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm : NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.
- [24] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.
- [25] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*. Wiley, New York, 2nd edition, 2000.

- [26] B. Efron and R.J. Tibshirani. *An introduction to the Bootstrap. Monographs on Statistics & Applied Probability*. Number 57. Chapman & Hall/CRC, New York, 1993.
- [27] J.P. Egan. *Signal detection theory and ROC analysis, Series in Cognition and Perception*. Academic Press, New York, 1975.
- [28] M. Ehrgott. *Multicriteria Optimization*. Springer, 2nd edition, 2005.
- [29] T.A. El-Mihoub, A.A. Hopgood, L. Nolle, and A. Battersby. Hybrid Genetic Algorithms: A Review. *Engineering Letters*, 13:124–137, 2006.
- [30] F.P. Espinoza, B.S. Minsker, and D.E. Goldberg. A Self Adaptive Hybrid Genetic Algorithm. *Genetic and Evolutionary Computation Conference (GECCO)*, 2001.
- [31] EUROCONTROL. Guidance material for short term conflict alert. Eurocontrol, 2007a. Available from http://www.eurocontrol.int/safety-nets/public/standard_page/stca_01.html.
- [32] EUROCONTROL. Specification for short term conflict alert (updated: Edition 1.0), 22 November 2007b. Available as http://www.eurocontrol.int/safety-nets/gallery/content/public/es_STCA-10.pdf.
- [33] EUROCONTROL. Mode S Technical Overview. 2008. Available as http://www.eurocontrol.int/msa/public/standard_page/modes_tech_overview.html.
- [34] EUROCONTROL. Requirements for civil aircraft. 2009. Available as http://www.eurocontrol.be/avionics/public/standard_page/16_Avionics_civil.html.
- [35] R.M. Everson and J.E. Fieldsend. Multi-objective optimisation of safety related systems: An application to short term conflict alert. *IEEE Transactions on Evolutionary Computation*, 10(2):187–198, 2006.
- [36] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874, 2006.
- [37] J. Fieldsend and R. Everson. ROC Optimisation of Safety Related Systems. In J. Hernández-Orallo, C. Ferri, N. Lachiche, and P. Flach, editors, *ROCAI 2004, part of the 16th European Conference on Artificial Intelligence (ECAI)*, pages 37–44, 2004.
- [38] J.E Fieldsend and R.M. Everson. Multi-objective optimisation in the presence of uncertainty. *IEEE Congress on Evolutionary Computation*, 1–3:243–250, 2005.
- [39] J.E Fieldsend and R.M. Everson. On the efficient use of uncertainty when performing expensive ROC optimisation. *IEEE Congress on Evolutionary Computation*, 1–8:3984–3991, 2008.
- [40] J.E. Fieldsend, R.M. Everson, and S. Singh. Using unconstrained elite archives for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 7(3):305–323, June 2003.

- [41] C.M. Fonseca and P.J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, 1993.
- [42] C.M. Fonseca and P.J. Fleming. An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation*, 3(1):1–16, 1995.
- [43] C.M. Fonseca and P.J. Fleming. Multiobjective optimization. In T. Baeck, D.B. Fogel, and Z. Michalewicz, editors, *Handbook on Evolutionary Computation*. IOP Publishing Ltd and Oxford University Press, 1997. Section C4.5.
- [44] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.
- [45] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.
- [46] C-K. Goh and K.C. Tan. A Competitive-Cooperative Coevolutionary Paradigm for Dynamic Multiobjective Optimization. *IEEE Transactions on Evolutionary Computation*, 13:103–127, 2009.
- [47] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.
- [48] D.E. Goldberg. Sizing populations for serial and parallel genetic algorithms. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 70–79. Morgan Kaufmann, 1989.
- [49] D.E. Goldberg and S. Voessner. Optimizing Global-Local Search Hybrids. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 220–228. Morgan Kaufmann, 1999.
- [50] D.J. Hand and R.J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45:171–186, 2001.
- [51] R. Hayward. *STCA ISS4 specification*. NATS Ltd., Operational Analysis Department.
- [52] R. Hayward and R.J. Howell. *Proposed Specification of Algorithms for an Enhanced Short Term Conflict Alert (ESTCA)*. NATS Ltd., Operational Analysis Department, issue 5 edition, September 2008.
- [53] J. Horn, N. Nafpliotis, and D.E. Goldberg. A Niche Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, 1994.

- [54] E.J. Hughes. Evolutionary Multi-objective Ranking with Uncertainty and Noise. In E. Zitzler et al, editor, *Evolutionary Multi-Criterion Optimization*, volume 1993 of *LNCS*, pages 329–343, 2001.
- [55] E.J. Hughes. Multiple Single Objective Pareto Sampling. *IEEE Congress on Evolutionary Computation*, pages 2678–2684, 2003.
- [56] E.J. Hughes. MSOPS-II: A general-purpose Many-Objective optimiser. *IEEE Congress on Evolutionary Computation*, pages 3944–3951, 2007.
- [57] Y. Jin. A Comprehensive Survey of Fitness Approximation in Evolutionary Computation. *Soft Computing*, 9:3–12, 2005.
- [58] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983.
- [59] J. Knowles. ParEGO: A Hybrid Algorithm with On-line Landscape Approximation for Expensive Multiobjective Optimization Problems. *IEEE Transactions on Evolutionary Computation*, pages 50–66, 2006.
- [60] J. Knowles and E.J. Hughes. Multiobjective Optimization on a Budget of 250 Evaluations. In *Evolutionary Multi-criterion Optimization. 3rd International Conference (EMO'05)*, pages 176–190. Springer, 2005.
- [61] J.D. Knowles and D.W. Corne. The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multiobjective Optimisation. In *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*, pages 98–105, 1999.
- [62] J.D. Knowles and D.W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolutionary Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.
- [63] K. Krishnakumar. Micro-Genetic Algorithms for Stationary and Non-Stationary Function Optimization. In *SPIE: Intelligent Control and Adaptive Systems*, pages 289–296, 1989.
- [64] M. Laumanns, E. Zitzler, and L. Thiele. On The Effects of Archiving, Elitism, And Density Based Selection in Evolutionary Multi-Objective Optimization. In *Lecture Notes in Computer Science, Evolutionary Multi-Criterion Optimization*, pages 181–196. Springer, 2001.
- [65] D.E. Moriarty and R. Miikkulainen. Formulating Neural Networks through Efficient and Adaptive Coevolution. *Evolutionary Computation*, 5:373–399, 1998.
- [66] NATS. Strategic plan for safety - 2004. pages 14–15. Available as <http://www.nats.co.uk/uploads/Strategic-Plan-Safety-v3.pdf>.
- [67] M.A. Potter and K.A. De Jong. Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. *Evolutionary Computation*, 8:1–29, 2000.

- [68] G.T. Pulido and C.A. Coello Coello. The Micro Genetic Algorithm 2: Towards On-Line Adaption in Evolutionary Multiobjective Optimization. In *Proceedings of the 2nd International Conference on Evolutionary Multi-criterion Optimization (EMO'03)*, pages 252–266. Springer-Verlag, 2003.
- [69] J.T. Richardson, M.R. Palmer, G. Liepins, and M. Hilliard. Some guidelines for genetic algorithms with penalty functions. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 191–197. Morgan Kaufmann Publishers, 1989.
- [70] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.
- [71] J.D. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms*, pages 93–100. Lawrence Erlbaum Associates, 1985.
- [72] K.I. Smith. *A Study of Simulated Annealing Techniques for Multi-Objective Optimisation*. PhD thesis, School of Engineering Computer-science and Mathematics, The University of Exeter, 2006.
- [73] K.I. Smith, R.M. Everson, and J.E. Fieldsend. Simulated Annealing and Greedy Search for Multi-objective Optimisation. *Submitted to European Journal of Operations Research*, 2009.
- [74] N. Srinivas and K. Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computing*, 2(3):221–248, 1994.
- [75] A. Suppapitnarm, K.A. Seffen, G.T. Parks, and P.J. Clarkson. A Simulated Annealing Algorithm for Multiobjective Optimization. *Engineering Optimization*, 33:59–85, 2000.
- [76] K.C. Tan, T.H. Lee, and E.F. Khor. Evolutionary Algorithms for Multiobjective Optimization: Performance Assessments and Comparisons. *Artificial Intelligence Review*, 17(4):253–260, 2002.
- [77] J. Teich. Pareto-Front Exploration with Uncertain Objectives. In E. Zitzler et al, editor, *Evolutionary Multi-Criterion Optimization*, volume 1993 of *LNCS*, pages 314–328, 2001.
- [78] D. Veldhuizen and G. Lamont. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, 8(2):125–147, 2000.
- [79] D. Whitley, V. Scott Gordon, and K. Mathias. Lamarckian Evolution, The Baldwin Effect and Function Optimization. In *Lecture Notes in Computer Science*, pages 6–15. Springer-Verlag, 1994.
- [80] Xin Yao, Yong Liu, and Guangming Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3:82–102, 1999.

- [81] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8:173–195, 2000.
- [82] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, 2002.
- [83] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, November 1999.
- [84] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V. Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.