Evolutionary Optimisation of Network Flow Plans for Emergency Movement in the Built Environment

Thomas Reginald French

Doctor of Philosophy
Centre for Intelligent Systems and their Applications
School of Informatics
University of Edinburgh
2011

Abstract

Planning for emergency evacuation, and, more generally, for emergency movement involving both evacuation (egress) of occupants and ingress of first responders, presents important and challenging problems. A number of the current issues that arise during emergency incidents are due to the uncertainty and transiency of environmental conditions. In general, movement plans are formulated at building design-time, and those involved, such as building occupants and emergency responders, are left to adapt routing plans to actual events as they unfold. In the context of next-generation emergency response systems, it has been proposed to dynamically plan and route individuals during an emergency event, replanning to take account of changes in the environment.

In this work, an emergency movement problem, the Maximal Safest Escape (MSE) problem, is formulated in terms that model the uncertain and transient environmental conditions as a flow problem in time-dependent networks with time-varying and stochastic edge travel-times and capacities (STV Networks). The objective of the MSE problem is to find flow patterns with the a priori maximal probability of successfully conveying all supply from the source to the sink in some given STV Network. The MSE and its deterministic counterpart are proved to be NP-hard. Furthermore, due to inherent complexity in evaluating the exact quality of candidate solutions, a simulation approximation method is presented based on well-known Monte-Carlo sampling methods.

Given the complexity of the problem, and using the approximation method for evaluating solutions, it is proposed to tackle the MSE problem using a metaheuristic approach based on an existing framework that integrates Evolutionary Algorithms (EA) with a state-of-the-art statistical ranking and selection method, the Optimal Computing Budget Allocation (OCBA). Several improvements are proposed for the framework to reduce the computational demand of the ranking method. Empirically, the approach is compared with a simple fitness averaging approach and conditions under which the integrated framework is more efficient are investigated. The performance of the EA is compared against upper and lower bounds on optimal solutions. An upper bound is established through the “wait-and-see” bound, and a lower bound by a naïve random search algorithm (RSA). An experimental design is presented that allows for a fair comparison between the EA and the RSA. While there is no guarantee that the EA will find optimal solutions, this work demonstrates that the EA can still find *useful* solutions; *useful* solutions are those that are at least better than some baseline, here the lower bound, in terms of solution quality and computational effort. Experimentally, it is demonstrated that the EA performs significantly better than the baseline. Also, the EA finds solutions relatively close to the upper bound; however, it is difficult to establish how optimistic the upper bounds. The main approach is also compared against an existing approach developed for solving a related problem wrapped in a heuristic procedure in order to apply the approach to the MSE. Empirical results show that the heuristic approach requires significantly less computation time, but finds solutions of significantly lower quality.

Overall, this work introduces and empirically verifies the efficacy of a metaheuristic based on a framework integrating EAs with a state-of-the-art statistical ranking and selection technique, the OCBA, for a novel flow problem in STV Networks. It is suggested that the lessons learned during the course of this work, along with the specific techniques developed, may be relevant for addressing other flow problems of similar complexity.

Acknowledgements

First, and foremost, I'd like to thank my supervisors: Austin Tate, Stephen Potter, Gerhard Wickler, Jano van Hemert, and Jose Torero. They have individually, and collectively, supported, taught, challenged, driven me, and much more — mostly willingly. Thank you.

Next I'd like to acknowledge the BRE Trust for their financial support, and specifically Debbie Smith at BRE as my industrial supervisor. Thank you for the opportunity to have been a part of the FireGrid project; may it become a reality some day.

To CISA, my gratitude for providing a challenging and supportive research environment, and also some welcome financial support. Thanks to all members of CISA, and wider, for their encouragement and support, in particular, friends and colleagues: Michael, Paolo, Adam, Adela, Areti, Lysimachos, Michael C, Omar, Jacques, Dave, Ewan. Also, the "Agent's Factory": George, Aris, Iain, Francesco, Matt, for their antics and for inventing the 'Science-O-Meter'. What does it say today? Additional thanks to those souls who tolerated sharing an office with me at one point or another — it must have been tough.

'Healthy body, healthy mind' — thanks to my squash partners: Jacques, Michael and Omar; our sessions were an important outlet for the frustrations of quotidian PhD life.

I'd also like to thank my examiners: Alan Smaill and Jurgen Branke, for a stimulating and challenging VIVA.

Dear friends (and colleagues), Cecilia and Adam: my heartfelt gratitude for your support and friendship (and tolerance), and for getting me interested in FireGrid in the first place way back when. And of course those numerous late-night conversations (debates? discussions? arguments?).

Maria: thank you for your support and for caring; more so for putting up with me in the (drawn-out) final stages.

Mum, Dad, Iain, and James: infinite thanks for your unflagging support throughout. This was my endeavour but I dragged you along. I dedicate this thesis to you.

I would also like to thank early teachers and collaborators at the Institute for Perception, Action and Behaviour: Barbara Webb, Richard Reeve and Darren Smith, for their support and encouragement during my initiation to research.

Onwards and upwards...

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Thomas Reginald French*)

Publications

Early work was presented at the Metaheuristics International Conference, 2009:
French, T., van Hemert, J., Potter, S., Wickler, G., and Tate, A. (2009) *An Evolutionary Algorithm for the Safest Weighted Escape Problem in Stochastic, Time-Varying Networks* in Metaheuristics International Conference, Hamburg, Germany

To mum and dad;
Iain and James

To make this dark world better and more bright.
Oh, with what joy and love I understand
These master-souls that ache for truth and light.
(exc. ‘Men of Science’, Alexander Search)

Table of Contents

Table of Contents **x**

List of Figures **xviii**

List of Tables **xxi**

Nomenclature **xxv**

1 Introduction **1**

1.1 Motivation 1

1.2 Problem Statement 3

1.2.1 Research Hypothesis 4

1.3 Thesis Contributions 4

1.4 Thesis Outline 5

2 Literature Review:
Network Optimisation **7**

2.1 Introduction 7

2.2 Online Decision-Support for Emergency Evacuation 8

2.3 Simulation Models for Evacuation Planning 9

2.4 Modelling Emergency Evacuation with Flow Networks 11

2.5 Network Optimisation for Evacuation Planning 11

2.5.1 Dynamic Flow Networks 13

2.5.2 Time-Varying Flow Networks 15

2.5.3 Stochastic Networks 17

2.5.4 Opasanon's Safest Escape Problem 19

2.5.5 Stochastic, Time-Varying Flow Networks 20

2.5.6 Conclusions 21

2.6 Summary 21

3 Literature Review: Evolutionary Algorithms **23**

3.1 Introduction 23

3.2 Algorithms 25

3.2.1 Solution Representation 27

3.2.2 Initialisation 27

3.2.3 Mating Selection 28

3.2.4 Crossover 30

3.2.5 Mutation 30

3.2.6 Evaluation 31

3.2.7 Replacement Strategy 32

3.2.8 Termination 33

3.2.9 Constraint Handling 33

3.3 EAs in Combinatorial Optimisation 34

3.3.1 EAs for Shortest Path Problems 34

3.3.2 EAs for Network Flow Problems 36

3.3.3 Conclusions 37

3.4 EAs in Noisy Environments 37

3.4.1 Impact of Noise on EAs 38

3.4.2 Noisy Fitness Functions 39

3.4.3 Schmidt's Framework: EAs and OCBA 43

3.4.4 Conclusions 45

3.5 EAs in Stochastic Combinatorial Optimisation 45

3.5.1 Introduction 45

3.5.2 Applications 47

3.5.2.1 Minimum Time Network Flow Problem 48

3.5.2.2 Maximal Dynamic Expected Flows Problem 50

3.5.3 Conclusions 51

3.6 Summary 51

4 Maximal Safest Escape Problem **53**

4.1 Notation and Preliminaries 53

4.1.1 Basic Definitions 53

4.1.2 Single Source, Single Sink 54

4.1.3 First-In-First-Out 54
4.1.4 Supply/Demand Model 55
4.1.5 Paths and Flows 55
4.1.6 Network States 56
4.1.7 Example 57
4.2 Route Selection Criteria 58
4.2.1 Status Quo 58
4.2.2 Performance Criteria in STV Networks 60
4.3 Maximal Safest Escape Problem 62
4.3.1 Problem Definition 62
4.3.1.1 Definition as an Expectation 62
4.3.2 Example 63
4.3.3 Problem Complexity: NP-hardness 64
4.3.4 Problem Combinatorics 66
4.3.4.1 Paths 66
4.3.4.2 Flow Patterns 67
4.4 Stochastic Bounds on Optimal Solutions 68
4.4.1 Interchange Relaxation 68
4.4.2 The Deterministic MSE Problem 70
4.4.3 Solving the Deterministic MSE Problem 71
4.4.3.1 Heuristics 71
4.4.3.2 Greedy Approach 72
4.5 Summary 75

**5 Candidate Solution Generation,
Evaluation and Ranking 77**

5.1 Candidate Solution Generation 77
5.1.1 Random Paths and Flows in STV Networks 78
5.1.2 Directed Acyclic Networks 78
5.1.3 Directed Cyclic Networks 80
5.2 Candidate Solution Evaluation 81
5.2.1 Simulation Approximation for Estimating Solution Quality . . 82
5.2.1.1 Approximation Issues 82
5.2.2 Precision of Simulation Approximation 83
5.2.3 Sampling Frame: Network States 84

5.2.4 Sampling Frame: Flow Pattern States 86
5.3 Comparing Candidate Solutions 89
5.4 Statistical Selection and Ranking 91
5.4.1 Optimal Computing Budget Allocation 93
5.4.2 Notation and Preliminaries 93
5.4.3 OCBA Allocation Rule 95
5.4.4 OCBA for Selecting the Best System 95
5.4.5 Stopping Rules for OCBA 96
5.4.6 Integrating OCBA with EAs 96
5.4.7 EA Order Information 97
5.4.7.1 Replacement Strategies 98
5.4.7.2 Selection Operators 99
5.4.7.3 Size of the Comparison Set 100
5.4.8 Using OCBA to Generate Order Information 100
5.4.9 The $\mathcal{OCBA}^{EA}_{\delta^*}$ Procedure 101
5.4.10 An Optimisation of and a Heuristic for the OCBA 102
5.4.10.1 Faster Evaluation of Thought Experiment 102
5.4.10.2 Heuristic for Faster Selection of Best q Individuals . 104
5.4.11 Informative Priors for Series of Selection Problems 105
5.4.12 $\mathcal{OCBA}^{EA}_{\delta^*}$ Parameters 106
5.5 Summary 108

6 Solution Approaches **111**
6.1 Noisy Evolutionary Algorithm 111
6.1.1 Candidate Solution Representation 113
6.1.1.1 Survey of Representations 113
6.1.1.2 MSE Representation 114
6.1.2 Initialisation 115
6.1.3 The Neighbourhood 116
6.1.3.1 Multiset Preliminaries 117
6.1.3.2 Neighbourhood function 118
6.1.4 Crossover 118
6.1.4.1 Multiset Operator 119
6.1.5 Mutation 120
6.1.5.1 Multiset Operator 121

6.1.6 Selection Strategies 121
6.1.6.1 Mating Selection 121
6.1.6.2 Replacement strategy 122
6.1.7 Evaluation 122
6.1.7.1 Noisy Fitness Function 122
6.1.7.2 Enhanced Noisy Fitness Function 123
6.1.7.3 $\mathcal{OCBA}_{\delta*}^{EA}$ for the MSE 128
6.1.8 Stopping Criteria 130
6.1.8.1 Final Selection of Best 131
6.1.9 NEA$_{MEAN}$ 133
6.1.10 NEA Parameters 133
6.2 Random Search Algorithm 133
6.2.1 Parameters 134
6.3 Summary 135

7 Experimentation and Analysis 137
7.1 Implementation 137
7.1.1 Problem Instance Generator 137
7.1.1.1 Graph Topology 138
7.1.1.2 Edge Properties 139
7.2 Experimentation 140
7.2.1 Experimental Design 141
7.2.1.1 Problem Classes 141
7.2.1.2 Performance Criteria 144
7.2.1.3 Experimental Procedure 145
7.2.2 Parameter Tuning 146
7.2.2.1 Experimental Procedure 147
7.2.2.2 NEA$_{OCBA}$ Parameters 147
7.2.2.3 Results and Analysis 148
7.2.3 Evaluation 154
7.2.3.1 Experimental Procedure 154
7.2.3.2 Results and Analysis 155
7.2.3.3 Validation of Results 157
7.3 Opasanon's Safest Escape Algorithm 157
7.4 Discussion 159

7.4.1 Parameter Tuning 159
7.4.1.1 NEA parameters vs. OCBA parameters 159
7.4.1.2 OCBA as part of the NEA 160
7.4.1.3 Effect of Target Confidence Level 161
7.4.1.4 Setting the Initial Sample Size 161
7.4.1.5 Summary 162
7.4.2 Evaluation 162
7.4.2.1 NEA Variants: *OCBA* and *MEAN* 163
7.4.3 Random Search Algorithm 166
7.4.4 Stochastic Upper Bound 166
7.5 Summary 167

8 Conclusions **179**
8.1 Summary 179
8.2 Main Contributions 180
8.3 Modelling Assumptions 182
8.3.1 Discrete time horizon 183
8.3.2 Edge property probability distributions 183
8.3.3 Network supply 184
8.3.4 Occupant modelling 184
8.3.5 Solution constraints 185
8.3.6 Sharing risk 186
8.4 Future Work 186
8.4.1 Network Model and Problem 187
8.4.2 Algorithms 187
8.4.2.1 Noisy Evolutionary Algorithms 187
8.4.2.2 Stochastic Upper Bound 188
8.4.3 Evaluation 189
8.5 Concluding Remarks 190

A Additional Results **191**
A.1 Parameter Tuning 191
A.2 Evaluation 191
A.2.1 Validation of Results 209
A.2.2 Evaluation of NEA$_{OCBA}$ with HSEA 220
A.3 Discussion 221

Bibliography **237**

List of Figures

4.1 Example STV Network Edge 57
4.2 Example STV Network: Topological Graph 58

5.1 Example graph 79

6.1 Normal Q-Q Plots of fitness distributions for a typical candidate solution in offspring population for respective generations. Taken from results of NEA$_{OCBA}$ for instance 13 of PC3 from the experimentation detailed in Section 7.2.3. 129

7.1 Parameter Tuning: SQ for PC1-PC4, C1-C8, UB 150
7.2 Parameter Tuning: SQ for PC5-PC8, C1-C8, UB 151
7.3 Parameter Tuning: CT for PC1-PC8, C1-C8 169
7.4 Parameter Tuning: N for PC1-PC8, C1-C8 170
7.5 Parameter Tuning: SQ for PC1-PC4, C4, C9-C11,UB 171
7.6 Parameter Tuning: SQ for PC5-PC8, C4, C9-C11,UB 172
7.7 Parameter Tuning: CT for PC1-PC8, C4, C9-C11 173
7.8 Parameter Tuning: N for PC1-PC8, C4, C9-C11 174
7.9 Evaluation: SQ for PC1-PC4 175
7.10 Evaluation: SQ for PC5-PC8 176
7.11 Evaluation: CT for PC1-PC8 177
7.12 Evaluation: N for PC1-PC8 178

A.1 Parameter Tuning: CT Pairwise Differences for PC1-PC8, C1-C8 (NB: y-axis scales vary, and grey boxes indicate significant differences at 0.05 level.) 198
A.2 Parameter Tuning: N Pairwise Differences for PC1-PC8, C1-C8 (NB: y-axis scales vary, and grey boxes indicate significant differences at 0.05 level.) 199

A.3 Parameter Tuning: SQ Pairwise Differences for PC1-PC8, C4, C9-C11 (NB: y-axis scales vary, and grey boxes indicate significant differences at 0.05 level.) 206
A.4 Parameter Tuning: CT Pairwise Differences for PC1-PC8, C4, C9-C11 (NB: y-axis scales vary, and grey boxes indicate significant differences at 0.05 level.) 207
A.5 Parameter Tuning: N Pairwise Differences for PC1-PC8, C4, C9-C11 (NB: y-axis scales vary, and grey boxes indicate significant differences at 0.05 level.) 208
A.6 Validation: SQ for PC1-PC4 209
A.7 Validation: SQ for PC5-PC8 220
A.8 Validation: CT for PC1-PC8 223
A.9 Validation: N for PC1-PC8 224
A.10 HSEA Evaluation: SQ for PC1-PC4 225
A.11 HSEA Evaluation: SQ for PC5-PC8 226
A.12 HSEA Evaluation: CT for PC1-PC8 227
A.13 HSEA Evaluation: Differences between HSEA and NEA$_{OCBA}$ (N.B. Grey boxes indicate significant results at 0.05 level.) 229
A.14 Discussion: SQ for PC1-PC4 230
A.15 Discussion: SQ for PC5-PC8 231
A.16 Discussion: CT for PC1-PC8 232
A.17 Discussion: N for PC1-PC8 233
A.18 Discussion: Differences between NEA$_{OCBA}$ and NEA$^{NS}_{OCBA}$ (N.B. Grey boxes indicate significant results at 0.05 level.) 235
A.19 Discussion: Differences between NEA$_{MEAN}$ and NEA$^{NS}_{MEAN}$ (N.B. Grey boxes indicate significant results at 0.05 level.) 236

List of Tables

2.1 Network Models 13

4.1 Example STV Network: Network Properties 58
4.2 Example STV Network: Flow Patterns (pairs with multiplicity of 0 are excluded for readability) 59
4.3 Example STV Network: Network Edge Capacities 59
4.4 Example STV Network: Network Edge Travel Times 59
4.5 Example STV Network: Flow Patterns with MSE Solution Quality . . 64

5.1 Precision of Sample Sizes 84
5.2 Size of comparison set for $(\mu+\mu)$ EA 101
5.3 $\mathcal{OCBA}_{\delta*}^{EA}$ Parameters 108

6.1 Final Selection Parameters 132
6.2 NEA Parameters 133
6.3 RSA Input Parameters that differ from NEA$_{OCBA}$ 134

7.1 Measured Properties of Generated Graphs 139
7.2 Fixed Problem Class Parameters 143
7.3 Variable Problem Class Parameters 143
7.4 Problem Classes 144
7.5 Experimental Parameters 146
7.6 Algorithm Input Parameters 146
7.7 Experimental Parameters for Parameter Tuning 147
7.8 Fixed Algorithm Input Parameters 148
7.9 Results from Pilot Studies 148
7.10 NEA$_{OCBA}$ Parameter Values for Parameter Tuning 149
7.11 NEA$_{OCBA}$ Configurations for Parameter Tuning 149

7.12 Additional NEA$_{OCBA}$ Configurations for Parameter Tuning (bold values indicate different values from C4) 153
7.13 Experimental Parameters for Evaluation 155
7.14 Algorithm Input Parameters for all Problem Classes (bold values are not shared) 155

A.1 Parameter Tuning: Results for SQ, PC1-PC4, C1-C8, UB 192
A.2 Parameter Tuning: Results for SQ, PC5-PC8, C1-C8, UB 193
A.3 Parameter Tuning: Results for CT (milliseconds), PC1-PC4, C1-C8, UB 194
A.4 Parameter Tuning: Results for CT (milliseconds), PC5-PC8, C1-C8, UB 195
A.5 Parameter Tuning: Results for N, PC1-PC4, C1-C8 196
A.6 Parameter Tuning: Results for N, PC5-PC8, C1-C8 197
A.7 Parameter Tuning: Results for SQ, PC1-PC4, C4, C9-C11 200
A.8 Parameter Tuning: Results for SQ, PC5-PC8, C4, C9-C11 201
A.9 Parameter Tuning: Results for CT (milliseconds), PC1-PC4, C4, C9-C11202
A.10 Parameter Tuning: Results for CT (milliseconds), PC5-PC8, C4, C9-C11203
A.11 Parameter Tuning: Results for N, PC1-PC4, C4, C9-C11 204
A.12 Parameter Tuning: Results for N, PC5-PC8, C4, C9-C11 205
A.13 Evaluation: Results for SQ, PC1-PC4 210
A.14 Evaluation: Results for SQ, PC5-PC8 211
A.15 Evaluation: Results for CT (ms), PC1-PC4 212
A.16 Evaluation: Results for CT (ms), PC5-PC8 213
A.17 Evaluation: Results for N, PC1-PC4 214
A.18 Evaluation: Results for N, PC5-PC8 215
A.19 Evaluation: P-values for post hoc pairwise comparison tests, SQ, PC1-PC4. (Bold indicates significant result at 0.05 level.) 216
A.20 Evaluation: P-values for post hoc pairwise comparison tests, SQ, PC5-PC8. (Bold indicates significant result at 0.05 level.) 217
A.21 Evaluation: P-values for post hoc pairwise comparison tests, CT, PC1-PC4. (Bold indicates significant result at 0.05 level.) 218
A.22 Evaluation: P-values for post hoc pairwise comparison tests, CT, PC5-PC8. (Bold indicates significant result at 0.05 level.) 219
A.23 Evaluation: P-values for post hoc pairwise comparison tests, N, PC1-PC8. (Bold indicates significant result at 0.05 level.) 222

A.24 HSEA Evaluation: P-values for hypothesis tests between the HSEA and NEA$_{OCBA}$. (Bold indicates significant result at 0.05 level.) 228
A.25 Discussion: P-values for hypothesis tests between NEA$^{NS}_{OCBA}$ and NEA$^{NS}_{MEAN}$. (Bold indicates significant result at 0.05 level.) 228
A.26 Discussion: P-values for hypothesis tests between the NEA$_{OCBA}$ and the NEA$^{NS}_{OCBA}$. (Bold indicates significant result at 0.05 level.) 234
A.27 Discussion: P-values for hypothesis tests between the NEA$_{MEAN}$ and the NEA$^{NS}_{MEAN}$. (Bold indicates significant result at 0.05 level. Hyphen indicates identical results.) 234

Nomenclature

$[T] \equiv \{0,1,\dots,T\}$	Time horizon of interest discretized into small time intervals, page 54
α^*	OCBA target confidence level: $1-\alpha^*$, page 108
$\bar{\mu}_{ij}(t)$	Random variable modelling capacity value for edge $(i,j) \in \mathscr{E}$ at time $t \in [T]$, page 57
$\bar{\tau}_{ij}(t)$	Random variable modelling travel time value for edge $(i,j) \in \mathscr{E}$ at time $t \in [T]$, page 57
$\bar{\xi}$	Random event modelling the state of network $\mathscr{N}$, page 57
$\bar{\xi}_\psi$	Random event modelling the state of flow pattern $\psi \in \Psi$, page 87
$\bar{n}_0$	OCBA initial sample size, page 107
$\beta^d_{ij}(t)$	Probability associated with capacity value $\mu^d_{ij}(t) \in \mathscr{M}$ for edge $(i,j) \in \mathscr{E}$ at departure time $t \in [T]$ and index $d \in \{1,\dots,D\}$, $\beta^d_{ij}(t) \in \mathscr{B}$, page 55
δ^*	OCBA size of the indifference-zone, page 108
$\Gamma^{+1}(i)$	Set of successor nodes of node $i \in \mathscr{V}$, page 55
$\Gamma^{-1}(i)$	Set of predecessor nodes of node $i \in \mathscr{V}$, page 55
$\hat{G}^n_\psi$	Monte Carlo estimate of solution evaluation for flow pattern ψ of sample size n, page 83
$\hat{U}$	Deterministic upper bound for optimal solutions to the MSE problem, page 71
$\hat{U}_n$	Stochastic upper bound for optimal solutions to the MSE problem , page 71
$\hat{v}$	Cardinality of $\mathscr{V}$ excluding source node s and sink node l, $\hat{v} = v-2$, page 67
κ	Maximum number of EA generations, page 131

λ	Number of individuals in EA offspring population, page 27
$\mathbb{P}$	Probability distribution function for network states Ξ, page 57
$\mathbb{P}_2$	Probability distribution function for flow pattern states, Ξ_ψ, for flow pattern $\psi \in \Psi$, page 88
$\mathcal{C}$	Set of comparison pairs for OCBA procedure, page 98
$\mathcal{F}_\psi$	Set of feasible states for $\psi \in \Psi$ given a set of problem constraints, page 57
$\mathcal{P}_i$	EA population of generation i, page 27
$\mathscr{B}$	The set of probabilities associated with edge capacities for network $\mathscr{N}$, page 55
$\mathscr{E}$	Set of edges (or arcs) in $\mathscr{G}$, page 54
$\mathscr{G}$	Finite directed graph, page 54
$\mathscr{M}$	The set of edge capacities for a network $\mathscr{N}$, page 55
$\mathscr{N}$	A time-dependent network, page 54
$\mathscr{R}$	The set of probabilities associated with edge travel times for a network $\mathscr{N}$, page 55
$\mathscr{T}$	The set of edge travel times for network $\mathscr{N}$, page 55
$\mathscr{V}$	Set of nodes (or vertices) in $\mathscr{G}$, page 54
μ	Number of individuals in EA population, page 27
$\mu_{ij}^{\xi}(t)$	Capacity value for edge (i,j) at departure time t for network state $\xi \in \Xi$, page 57
$\mu_{ij}^{d}(t)$	Capacity value for edge $(i,j) \in \mathscr{E}$ at departure time $t \in [T]$ with index $d \in \{1,\ldots,D\}$, $\mu_{ij}^{d}(t) \in \mathscr{M}$, page 55
Ω	Set of paths in network $\mathscr{N}$, page 56
Ψ	Set of flow patterns in network $\mathscr{N}$, page 57
ψ	Flow pattern in network $\mathscr{N}$, page 56
ψ^*	Optimal solution for MSE problem for network $\mathscr{N}$, page 64

$\rho_{ij}^{h}(t)$	Probability associated with travel time value $\tau_{ij}^{h}(t) \in \mathscr{T}$ for edge $(i,j) \in \mathscr{E}$ at departure time $t \in [T]$ and index $h \in \{1,\ldots,H\}$, $\rho_{ij}^{h}(t) \in \mathscr{R}$, page 55
σ	Directed simple $s-l$ path in the graph $\mathscr{G}$, page 56
$\tau_{ij}^{\xi}(t)$	Capacity value for edge (i,j) at departure time t for network state $\xi \in \Xi$, page 57
$\tau_{ij}^{h}(t)$	Travel time value for edge $(i,j) \in \mathscr{E}$ at departure time $t \in [T]$ with index $h \in \{1,\ldots,H\}$, $\tau_{ij}^{h}(t) \in \mathscr{T}$, page 55
θ	OCBA number of additional samples for each selected individual, page 108
Ξ	The set of states of network $\mathscr{N}$, page 57
ξ	A realised state of network $\mathscr{N}$, $\xi \in \Xi$, page 57
ξ_{ψ}	A realisation of a flow pattern state for flow pattern $\psi \in \Psi$, $\xi_{\psi} \in \Xi_{\psi}$, page 87
Ξ_{ψ}	Set of flow pattern states for flow pattern $\psi \in \Psi$, page 87
B	Total supply to the network: $B = \sum_{t=0}^{T} b_s(t)$, page 56
$b_i(t)$	Supply/demand for node $i \in \mathscr{V}$ at departure time $t \in [T]$, page 56
D	Number of capacity values associated with each edge $(i,j) \in \mathscr{E}$ and each departure time $t \in [T]$, page 55
e	Cardinality of the set $\mathscr{E}$, page 54
$G(\psi)$	MSE objective function value for flow pattern $\psi \in \Psi$, page 64
H	Number of travel time values associated with each edge $(i,j) \in \mathscr{E}$ and each departure time $t \in [T]$, page 55
$h(\psi,\xi)$	Indicator function for set of feasible network states $\mathcal{F}_{\psi}$ for flow pattern $\psi \in \Psi$, given network state $\xi \in \Xi$, page 64
$I_X(\xi)$	Indicator function for set of network states $X \subseteq \Xi$ given network state $\xi \in \Xi$, page 63
l	Sink node of network $\mathscr{N}$, page 55
p_c	EA crossover rate, page 27

p_m	EA mutation rate, page 27
q	OCBA number of individuals selected each stage for allocation of additional samples, page 108
S	The set of departure times $t \in [T]$ from the source s where $b_s(t) > 0$, page 56
s	Source node of network $\mathscr{N}$, page 55
U	Deterministic upper bound for optimal solutions to the MSE problem, page 70
U_n	Stochastic upper bound for optimal solutions to the MSE problem , page 70
v	Cardinality of the set $\mathscr{V}$, page 54
$x_{ij}(t)$	The flow on edge $(i, j) \in \mathscr{E}$ departing node $i \in \mathscr{V}$ at time $t \in [T]$, page 55
z^*	Value of optimal solution, $G(\psi^*)$, for MSE problem for network $\mathscr{N}$, page 64

Chapter 1

Introduction

"Quo Vadimus?"

1.1 Motivation

Effective emergency response is often hampered by a lack of information about an unfolding incident. This is mainly due to the highly transient and uncertain nature of such events. For example, the fire brigade may attend an incident in a building without knowledge of the source or state of the fire, or even the occupancy of the building. Clearly, this uncertainty has had a huge impact on the present methods and procedures used by emergency responders.

Projects investigating next-generation emergency-response systems, like FireGrid (Berry et al., 2005; Upadhyay et al., 2008; Han et al., 2010), currently focussed on emergencies in the built environment, are attempting to change the handling of incidents. A key aspect of these projects involves the development of buildings that are 'intelligent', in that they are proactive in monitoring and responding to events in their environment. In the case of an emergency, these buildings will have the capacity to detect and recognise an event, like a fire, and react accordingly by, for example, activating first-line responses like sprinklers and fire alarms, and manipulating ventilation to mitigate smoke spread. Furthermore, it is envisioned that buildings will be linked to a wider infrastructure enabling them to escalate an incident to external emergency responders, should it be necessary. The building system would then assist responders by continually providing up-to-date information about unfolding events, like the spread of fire and smoke, the structural integrity of the building, and the whereabouts of occupants. Coupled with information services and analytical tools at the responders' disposal, including decision-support systems and simulation tools, this should enable the responders to observe and predict the progression of an incident more clearly and, thus, help them make more informed and better decisions.

An important part of emergency response is the evacuation of building occupants. Currently, emergency evacuation plans are developed during the design phase of a building, often using computer simulation to evaluate how humans might behave given numerous scenarios. This process involves the consideration of a number of factors, including, for example, occupancy and occupant knowledge of a building. In general, when implemented the evacuation plans are then communicated to occupants by way of a range of media, including the use of sound, signs and lighting, and through documentation like route maps. However, in the event of an actual emergency occupants are left to adapt the plans to the unfolding events of which they most likely have ambiguous and uncertain knowledge. In reality, the routes they choose, which may or may not agree with the emergency plans, could be obstructed, congested or even, in the worst case, impassable.

Using an 'intelligent' building's real-time and enriched view of an incident and knowledge of the whereabouts of its occupants, it may be possible to plan the safest escape route for individual occupants, and monitor and respond to the execution of this plan, adapting it to the changing environment by, for example, re-routing evacuees to avoid untenable areas or prevent congestion. Communication of plans might be done using a variety of existing technologies, like dynamic signs and lighting, and public announcement systems. Incorporating the evacuation system into the wider intelligent building infrastructure would allow other response activities to be adjusted appropriately to adapt to the evacuation plan. It would also enable information to be provided to emergency responders about the current state of evacuation, and, importantly, instructions to guide them to, for example, the source of the fire and occupants who have taken refuge or are unable to evacuate due to injury.

Furthermore, movement plans are currently developed using *time* as the performance criterion, such as total or last evacuation time (SFPE, 2002). Here time acts a simple surrogate for risk; it is used as a heuristic for measuring potential exposure to harm, and so, the argument goes, by minimising the time of potential exposure to a hazard, the risk of harm is also minimised. While this has proven a useful approach to date, in the context of future emergency response systems, it may be possible to model the transiency and uncertainty of a hazardous environment more directly, and, hence, provide a more accurate understanding of the risks posed. For example, it is not necessarily the case that the shortest or fastest path, in terms of *time*, is the safest. In other words, a more direct approach could provide a better informed assessment of the risk to individuals, such as building occupants and emergency responders; an assessment which can then be used for determining the safest movement plans.

The goal of this project is to investigate methods for finding safe emergency movement plans in dynamic and uncertain environments, specifically the built environment.

1.2 Problem Statement

Motivated by this vision of future emergency-response systems, a formal model of the built environment is developed herein that attempts to capture the transient and uncertain environmental conditions. For the problem of evacuation planning within this context, an optimisation problem, called the Maximal Safest Escape Problem, is defined with the goal of finding evacuation plans that have *the highest likelihood of successfully supporting the egress of occupants from their initial locations to places of safety in transient and uncertain environments*.

The problem is formally defined in time-dependent flow networks with time-varying, stochastic, edge travel times and capacities. In these networks, building circulation systems are represented using nodes and edges, where nodes model distinct locations such as doors and corridor intersections, and edges, which connect nodes, represent spaces which people traverse, for example, corridors and stairwells. Occupants are represented homogeneously as individual units of supply that flow through the network. Node and edge properties can be used to represent building dimensions and properties, for example, edge capacities represent the number of units of supply that begin to traverse an edge at a particular point in time. In the networks, commonly two special types of node are identified, namely sources from which network supply originates, and sinks, destination nodes toward which supply moves. To model the transient and uncertain environmental conditions, edge properties are defined as time-varying and stochastic.

Given the network model, the Maximal Safest Escape Problem is defined as the problem of finding a routing plan that will convey all network supply from the sources to the sinks with the highest overall likelihood of successful traversal, given all potential realisations of network stochastic elements. The computational complexity of the problem and its deterministic counterpart are shown to be NP-hard, and thus it is unlikely that efficient, exact algorithms will be proposed for solving either problem.

Given the complexity results, the goal of this work is therefore to develop approximate procedures, based on Evolutionary Algorithms, that efficiently find high-quality, but not necessarily optimal, solutions, termed *useful* solutions, to the Maximal Safest Escape Problem. Since optimal solutions are in general unknown, useful solutions are defined here as those that are better than some baseline in both solution quality and in computation time.

1.2.1 Research Hypothesis

In support of this investigation to find an effective approach for solving the Maximal Safest Escape Problem, the research hypothesis driving this work is:

> **Evolutionary Algorithms will find *useful* solutions to the Maximal Safest Escape Problem in time-dependent flow networks with stochastic, time-varying edge capacities and travel times.**

Once again, the complexity of the problem indicates that in general it is unlikely that efficient, exact solution approaches will be proposed. Given this, herein high-quality, approximate solutions are desired, specifically those that are *useful*, that is, not necessarily optimal but are of high-quality and that are found within reasonable time. Here such a baseline on solution quality and computation time is established by a random search algorithm, and, additionally, stochastic upper bounds on optimal solutions are also provided for comparison.

1.3 Thesis Contributions

Towards the end of evaluating the given research hypothesis, the main contributions of the thesis are summarised as follows:

- Definition of a novel flow problem, the Maximal Safest Escape Problem (MSE), in time-dependent flow networks with stochastic, time-varying edge capacities and travel times.
- Formal computational complexity results for the MSE and its deterministic counterpart.

- Extension of a Statistical Ranking and Selection procedure, the Optimal Computing Budget Allocation (OCBA) method, as part of a state-of-the-art framework integrating EAs and the OCBA applied here to the MSE, with:
  - an optimisation to improve the computational efficiency;
  - a heuristic to further improve the efficiency;
  - the use of informative Bayesian priors.
- Design and implementation of an Evolutionary Algorithm (EA) for solving the MSE.
- Empirical evaluation of the application of the EA to the MSE against a competing EA, a random search algorithm, stochastic upper bounds, and an existing approach developed for a related problem.

These contributions will be described in detail in the subsequent chapters.

1.4 Thesis Outline

The rest of the thesis has the following structure:

Literature Review: Network Optimisation Chapter 2 looks at the domain of interest, providing a review of approaches used for evacuation planning, in particular focussing on those works applying network flow models or closely related models to emergency evacuation planning. It begins with simple dynamic network models and progressively surveys more complex models, including those with both time-varying and stochastic network elements.

Literature Review: Evolutionary Algorithms Chapter 3 focusses on the solution approach of choice: Evolutionary Algorithms. It begins with an overview of Evolutionary Algorithms and briefly describes the components of a generic EA. A number of related areas are then surveyed, including the application of EAs to related shortest path and flow problems. Then it reviews both theoretical and applied works studying EAs in noisy/stochastic environments more generally, in particular focussing on a state-of-the-art framework integrating EAs with a Ranking and Selection procedure that is applied to the problem tackled herein.

Maximal Safest Escape Problem Chapter 4 provides a formal definition of stochastic, time-varying flow networks (STV Networks), and defines the problem studied herein, the Maximal Safest Escape Problem. Furthermore, it details the computational complexity of the problem. Finally, it describes a stochastic upper bounding procedure for the MSE.

Candidate Solution Generation, Evaluation and Ranking Chapter 5 is focussed on candidate solutions to the MSE; in particular, detailing methods for generating approximately equiprobable candidates, describing how to evaluate the solution quality of candidate solutions for the MSE using a simulation approximation approach, and finally detailing a state-of-the-art framework integrating EAs and a statistical Ranking and Selection (R&S) procedure for tackling problems in stochastic environments that is applied to the MSE. An optimisation and heuristic extension are proposed for improving the computational complexity of the R&S procedure.

Solution Approaches Chapter 6 details the proposed approaches for solving the MSE. It also describes two further approaches: an EA using a simple averaging approach to evaluate candidate solutions, and a random search algorithm.

Experimentation and Analysis Chapter 7 details the empirical evaluation of the proposed approaches for solving the MSE. It highlights some important implementation details, in particular the problem instance generator, and the experimentation carried out. The chapter closes with a discussion of the results and their implications.

Conclusions The final chapter (8) of the thesis details the conclusions of the work and identifies a number of avenues for future research.

Chapter 2

Literature Review: Network Optimisation

The following two chapters survey areas of the literature related to the research problem addressed in this thesis. This chapter provides the wider context of the work described herein, focussing on applications of network optimisation models to evacuation planning. The following chapter moves into a discussion of the general approach applied to the problem described herein, and in particular closely related works.

2.1 Introduction

This chapter begins by mentioning works in the area of online decision support for emergency evacuation for the built environment, in particular the FireGrid project. It also provides a brief survey of work in the area of simulation models for evacuation planning. This is followed by a discussion of a particular macroscopic branch of modelling involving flow networks, beginning with a discussion on how network models are used to represent building environments. Then, a review of works using network optimisation to solve evacuation problems in dynamic flow networks is given; specifically, works that directly address real-world evacuation planning problems and those that tackle evacuation inspired problems. This is followed by a review of works that use more elaborate network models, for example, those including time-varying network attributes like travel times, and those that include stochastic elements. Also mentioned are a few works that address operational evacuation planning.

2.2 Online Decision-Support for Emergency Evacuation

Several works, for example (Jones and Bukowski, 2001; Jones et al., 2005; Davis, 2007; Berry et al., 2005; Miller-Hooks and Krauthammer, 2007; Miller-Hooks, 2009), have proposed the idea of online decision support for emergency response with the ultimate goal of improving the management and efficiency of the handling of such hazardous scenarios. One aspect of the general response involves the safe and efficient movement of individuals involved, that is, the building evacuation (egress) of occupants and the managed ingress of responders. These works have conceptualised the idea, expounded the requirements, and discussed potential technological solutions.

A key idea of these projects involves the development of buildings that are 'intelligent' in the sense that they will be proactive in monitoring and responding to events within and around them. In the case of an emergency, these buildings will have the capacity to detect and recognise an event, such as a fire, and react accordingly by, for example, activating first-line responses like sprinklers and fire alarms, and manipulating ventilation to mitigate the effects of smoke. Furthermore, it is envisioned that buildings will be linked to a wider infrastructure enabling them to escalate an incident to external emergency responders, should it be necessary. The building system would then assist responders by continually providing up-to-date information about unfolding events, such as the spread of fire and smoke, the structural integrity of the building, and the whereabouts of occupants. The data could be collected through, for example, building sensor networks and video cameras. When coupled with information services and analytical tools such as decision-support systems and simulation tools, this could enable the emergency responders to observe and predict the progression of an incident more clearly and, thus, help them make more informed and better decisions.

Recently, the FireGrid project (Berry et al., 2005; Upadhyay et al., 2008; Han et al., 2010) designed and prototyped a system that provides the necessary infrastructure and systems for online decision support for emergency response in the built environment. The FireGrid project proposed a novel system architecture for improving the range and quality of information available to emergency responders, in the first instance for firefighters. The project required a system with the capabilities to capture information in real-time, to interpret the information accurately, and to present it in an accessible and concise manner. In order to accomplish this, an integrated architecture was designed, including sensors for detecting, among other things, smoke and heat, which feed data into complex computational models of fire and smoke spread, running on High Performance Computing resources accessed via Grid technology, and used for the prediction of major events such as structure collapse. A knowledge-based system was also developed for reasoning about the current environmental conditions and the forecasts in order to make accurate and concise interpretations for presentation to the responders so as to provide decision-support information about the unfolding incident.

Another important aspect of the safe handling of such hazardous scenarios concerns the safe egress of building occupants and the managed ingress of responders. Towards this end, using a computational framework of the sort proposed by the FireGrid project and knowledge of the whereabouts of its occupants, it may be possible to plan the safest escape route for individual occupants, and monitor and respond to the plan execution by adapting it to the changing environmental conditions. Plan changes could include re-routing evacuees to avoid untenable areas or to prevent congestion. The activities performed by building response systems might be adjusted appropriately to the plan. Additionally, these systems could provide information to emergency responders about the current state of evacuation, and, importantly, instructions to guide them, for example, to the source of the fire and to occupants who have taken refuge or who are unable to move due to injury. This is the vision that motivates the work described in this thesis.

2.3 Simulation Models for Evacuation Planning

Over the years, a number of studies have been carried out with the goal of understanding human behaviour during an emergency evacuation. This research, summarised in (SFPE, 2002), has provided a basis for computer simulation models for predicting the movements of occupants during egress in order to, for example, evaluate building designs against a number of potential scenarios. The focus of these models lies in estimating building emergency movement times (in particular the Required Safe Egress Time (RSET) measure) and, sometimes, in identifying issues such as potential congestion bottlenecks. The models have developed significantly over the years and are now considered an important tool in performance-based fire safety analysis.

From among this myriad of models, in general, two types of approach can be distinguished, namely those based on micro- and those on macroscopic models. In microscopic models, evacuees are considered as separate entities with individual characteristics, for example, they are given dimensions, walking speeds and memory. These details are used to determine an individual's movement decisions, such as which walkway to choose or whether to change direction. Microscopic models intend to model humans at a fine granularity and hence purport to make accurate claims about how people may evacuate a building. On the other hand, macroscopic models take a higher-level approach and model occupants as a homogeneous group, and hence do not consider differences among individuals. The general purpose of such models, which are mainly based on optimisation approaches, is to provide lower bounds on evacuation time. A common approach, which is adopted here, is to use flow network models for modelling and optimising emergency evacuation. For extensive surveys on simulation models, both micro- and macroscopic, see, for example, (Kuligowski and Peacock, 2005; Gwynne et al., 1999; Tjandra, 2003).

It is important to note that the majority of these models, both micro- and macroscopic, are used to predict human movement in the evaluation of potential building designs, and also to investigate incidents post hoc in order to gain insight into possible events with respect to emergency evacuation. They are, however, not used in situ, running in real-time to adapt egress plans during an actual evacuation as an incident unfolds, or providing up-to-date information to emergency responders to help in their response. In other words, they provide an off-line tool for analysis, not for operational planning.

Furthermore, currently movement plans are developed using *time* as the decisive parameter, such as total or last evacuation time (SFPE, 2002). Here *time* acts as a uni-dimensional surrogate for risk. It is used as a heuristic in measuring potential exposure to harm, and in general by defining the goal to minimise the time of potential exposure to a hazard, the risk of harm is also minimised. While this has proven a useful approach to date, in the context of future emergency response systems, e.g. FireGrid, it may be possible to model the transiency and uncertainty of a hazardous environment more directly, and, hence, provide a more accurate understanding of the risks posed. As a simple example, it is not necessarily the case that the shortest or fastest path, in terms of *time*, is the safest. In other words, a more direct approach could provide a better informed assessment of the risk to involved individuals, which can then be used for determining the safest movement plans.

The focus of this literature review is on macroscopic approaches, particularly the application of flow networks for evacuation planning, both in the design phase and for operational purposes.

2.4 Modelling Emergency Evacuation with Flow Networks

Flow-network based macroscopic approaches use an abstract network representation of a building's means of egress or circulation systems. These networks consist of a number of *nodes*, or *vertices*, connected by *edges*, or *arcs*.

In these models, in general, nodes in the network represent significant locations in a building, such as rooms, building exits and corridor intersections. The exact number of nodes in a model depends on a number of factors, including the particular building system, the granularity required for the model, and also the modeller's choices of "significant locations" in the building. The nodes are connected by edges, which represent paths connecting these locations, such as corridors and stairwells. Occupants are modelled as homogeneous abstract units, where typically one unit represents one person, and units are assumed not to interact directly but only implicitly through their common use of paths of limited capacity. Often each edge will be given certain properties that represent its state, such as a capacity and a traversal time. The capacity of an edge is typically represented by the number of units that can traverse that edge per unit of time. Traversal time is defined as the number of time units required to travel the length of an edge, that is, to get from the head node to the target node of the edge. Additionally, special nodes are often identified in the network, such as source nodes, that represent locations where occupants begin, and sink nodes, that represent locations of safety, such as refuges and fire exits towards which the occupants must move.

Given the importance of *time* as the main decisive parameter during emergency evacuation these network models often include a temporal dimension as well. This is discussed in detail below in Section 2.5.1. The specific optimisation to be performed depends on the particular problem definition; however, often when applied to evacuation the problems relate to minimising the time to evacuate building occupants, such as total time or last time to exit (SFPE, 2002). More details on network modelling of building circulation systems at the macroscopic level can be found in, for example, Tjandra (2003) and Miller-Hooks (2009).

We now review studies using network optimisation for evacuation planning.

2.5 Network Optimisation for Evacuation Planning

A multitude of works have investigated flow problems in static flow networks (SFNs). For a comprehensive introduction to the theory of flow networks and the details of a number of applications, see Ahuja et al. (1993). SFNs are, however, somewhat limited as models of real problems because they do not incorporate a temporal dimension. Without this dimension, SFNs cannot capture the evolution of a system over time; they are limited to representing the movement of units in a single wave. Fundamentally, they are unable to model the movement of flow over time or the transiency of network edge properties. Time is clearly an important consideration when modelling many real-world problems, and in particular evacuation planning where incidents are inherently transient and in which time is the decisive parameter for evacuation. It is due to these issues that dynamic networks, rather than static, have been more commonly applied to the problem of evacuation modelling and planning.

Dynamic flow networks are network models that include a temporal dimension, such that flow moves through the network over time. In this case, travel times dictate the number of time units flow takes to traverse an edge and capacities restrict the rate of flow on an edge. For extensive surveys of dynamic flow networks and their applications in general, see, for example, Aronson (1989); Powell et al. (1995); Kotnyek (2003); Lovetskii and Melamed (1987), and for a comprehensive survey of the mathematical modelling of evacuation problems using flow networks, see Hamacher and Tjandra (2002a). Due to the availability of extensive reviews, we only discuss works that consider flow networks relevant for modelling evacuation or that address evacuation planning directly. Also, this review focusses on models with finite, discrete time horizons.

Table 2.1 provides a list of the main network models addressed in this survey, highlighting the differences between them. Using a Y for 'yes' and a N for 'no', the table indicates the properties of network elements included in the various models, where:

D Temporal dimension is included but network properties are time-invariant.

TV Time-varying network properties.

S Stochastic but time-invariant network elements.

STV Stochastic and time-varying network elements.

Note that stochastic networks includes a number of different models, e.g. flow networks and queuing networks, which are united through their inclusion of stochastic elements. The following two entries: Safest Escape Problem and Stochastic, Time-Varying FNs are subclasses of the general Stochastic Networks class; however, they are included separately due to the relevance to the problem addressed herein. Also, the Safest Escape Problem is not a network model but it is included because it presents a flow problem for a specific network model that includes both deterministic and stochastic elements, and is particularly relevant to the problem addressed herein.

Table 2.1: Network Models

Model	D	TV	S	STV	Section
Static FNs	N	N	N	N	2.5
Dynamic FNs	Y	N	N	N	2.5.1
Time-Varying FNs	Y	Y	N	N	2.5.2
Stochastic Networks	Y & N	Y & N	Y	N	2.5.3
Safest Escape Problem	Y	Y	Y & N	Y & N	2.5.4
Stochastic, Time-Varying FNs	Y	Y	Y	Y	2.5.5

2.5.1 Dynamic Flow Networks

A number of studies address evacuation in some capacity using models based on dynamic flow networks. They vary in a number of ways: in the network model specifications, in the specific problems addressed and in the solution approaches. However, in general, two different flow problems in dynamic networks have been considered for evacuation planning. The two basic problems are:

1. Quickest Flow Problem: given a fixed supply, convey all the flow from source to sink in the minimum time possible. This is seen as a model of the classic evacuation problem of planning for the egress of all of some number of occupants to safety as fast as possible.

2. Earliest Arrival Flow Problem[1]: this problem involves sending the maximum flow across a network within some time bound T and for any $T' < T$. This is considered appropriate when the number of occupants of a building is unknown beforehand; for example, within a shopping centre.

[1]There is some inconsistency in the naming of flow problems, particularly the earliest arrival problem (for more discussion on this matter, see Kotnyek (2003)). Sometimes this problem is also referred to as the Universal Maximum Flow problem (UMF), for example, by Ford and Fulkerson (1962). However, UMF can also refer to the problem of maximising both the amount of flow leaving the source and flow arriving at the sink at every interval $[t, T]$, $0 \leq t \leq T$; for example, by Fleischer and Tardos (1998) and Orda and Rom (1995).

An early application of networks specifically to the evaluation of emergency paths was carried out by Berlin (1978). A network representation of building designs was used in order to evaluate escape potential, defined as the number of possible escape routes from a point in a building to any of the points of safety. Escape potential was judged to provide a reasonable indication of an individual's chance of selecting a path to a location of safety and, thus, an objective measure of the differences in safety between building designs.

In 1982, Chalmet et al. (1982) studied various flow networks models for building evacuation, and introduced the idea of using networks to represent building circulation systems. For this purpose, they argue that dynamic flow models are essential and suggest that the triple optimisation result of Jarvis and Ratliff (1982) could be useful as the optimisation objective. Briefly, the *triple optimisation* result highlights the equivalency between optimising the minimum evacuation time of the last unit to egress, the earliest arrival flow, and the minimum average time to evacuate all occupants. Chalmet et al. use the flow models to study building designs to identify potential bottlenecks.

Choi et al. (1984) investigated a building evacuation problem in dynamic flow networks with flow-dependent capacities on edges, looking for the maximum occupancy of a building such that in the case of an emergency evacuation all occupants can exit within a given time horizon: the *maximum dynamic flow problem*. They showed that an optimal solution exists given certain assumptions about the capacity function.

Hamacher and Tufekci (1987) studied the lexicographic[2] minimum cost flows problem for evacuation modelling. They showed how to prevent unnecessary movement within a building while optimising evacuation time using a multi-level priority system. They described how to evacuate a building divided into prioritized zones; a valid solution must evacuate the highest priority zone as quickly as possible, then the next highest zone as quickly as possible, and so on.

Choi et al. (1988) investigated three flow problems in dynamic flow networks with constant capacity values in the context of evacuation: 1) the maximum dynamic flow problem; 2) the minimum turnstile cost problem (triple optimization results: also minimises building evacuation time and maximised the cumulative number of people exiting the building for all time periods); 3) the minimax bottleneck problem (minimise time last person exits). For some specially structured networks greedy approaches are proposed, but for more general cases a linear programming method is suggested.

[2]Lexicographic flow problems are defined with a prioritised ordering on sources from which to ship supply.

Burkard et al. (1993) addressed the quickest flow problem in dynamic networks with single source and single sink. Polynomial and strongly polynomial algorithms for the quickest flow problem were developed.

Hoppe and Tardos (1994) then presented the first polynomial-time algorithms for two flow problems: the quickest flow problem with a fixed number of sources and sinks, and the lexicographic maximum dynamic flow problem; and a fully-polynomial approximation scheme for the earliest arrival flow problem. Later, Hoppe and Tardos (1995) proposed polynomial-time algorithms for the quickest transshipment problem, which extends the quickest flow problem (the evacuation problem) to multiple sources and multiple sinks. Fleischer and Tardos (1998); Fleischer (1998) extended this work and tackled many flow problems in continuous-time domains that had only been previously studied with discrete time models; many of their results also apply to the discrete time models. Kamiyama et al. (2009) improve on the polynomial time complexity of the algorithms proposed by Hoppe and Tardos (1995) for the evacuation problem but in a sub-class of problems, those with uniform path-lengths.

Later, Baumann and Skutella (2009) addressed the problem of the earliest arrival flows problem with multiple sources in dynamic networks with continuous time (although they note that their results also hold for the discrete model). They provided a strongly polynomial algorithm in the input plus output size of the problem.

Until now, the reviewed works have assumed dynamic network models with time-invariant network properties, such as edge travel times and capacities, and network supply; that is, the properties themselves are not functions of time but remain constant. The term *dynamic* refers only to the time-varying nature of the flow. However, the time-varying or time-dependent nature of network properties (edge travel times, capacities, and supply) is an important consideration for evacuation modelling, given the inherently transient conditions of an evacuation scenario. We now review works that incorporate these properties into their models.

2.5.2 Time-Varying Flow Networks

A host of works, including (Anderson et al., 1982; Philpott, 1990; Pullan, 1993; Orda and Rom, 1995; Pullan, 1997), have considered flow problems in dynamic networks where time is modelled continuously, and edge capacities, storage capacities (waiting) or costs are time-varying. However, these approaches do not apply when travel times are also considered time-varying, which, as indicated earlier, can be important in modelling evacuation scenarios.

Cai et al. (2001a,b) provided pseudopolynomial algorithms[3] for earliest arrival flow problems and mininum cost flow problems in dynamic flow networks with network properties, namely edge travel times and capacities, as discrete functions of time. They solve three variants of each problem, classified by their node waiting policy: allowing arbitrary waiting, waiting prohibited, and bounded waiting.

In (2003), Tjandra addressed several flow problems, including maximum dynamic flow problems, earliest arrival flow problems and quickest flow problems (including the evacuation problem) in dynamic networks with time-dependent edge capacities. Solution approaches are developed for each problem with pseudopolynomial time complexity, and the earliest arrivals model is applied to an evacuation case-study.

Miller-Hooks and Patterson (2004) provide pseudopolynomial algorithms for several quickest time problems in dynamic networks with time-varying network properties. They solve several flow problems in these networks, including the time-dependent quickest flow problem (single source, single sink), the time-dependent evacuation problem (multi-source, single-sink) and the quickest transshipment problem (multi-source, multi-sink). They also describe an efficient procedure for converting multiple-source, multiple-sink dynamic networks to equivalent single-source, single-sink dynamic networks with time-varying properties.

Building on the work in time-varying dynamic network problems, Lin et al. (2008) propose a multi-stage time-varying quickest flow approach for the optimisation of evacuation planning. Included in their model is the notion of *phased evacuation*, implemented through the use of priorities for groups that are released from the source according to the priority level. Algorithms from Hamacher and Tjandra (2002b) and Miller-Hooks and Patterson (2004) are applied after modification to handle the multi-stage problem formulation.

Looking towards the goal of providing approaches for operational purposes as discussed previously (Section 2.2), rather than solely evacuation pre-planning, in the context of, for example, future emergency response systems, Chen and Miller-Hooks

[3]Pseudopolynomial time algorithms are members of the class of exponential time algorithms; however, for certain bounded inputs they run in polynomial time. This is due to the fact that they run in polynomial time in the numeric value of an input parameter, e.g. T, rather than in the size of the input parameter, which would be $log(T)$ (roughly the number of bits required to store T). Since $O(T) \neq log(T)$, the algorithm cannot run in polynomial time in the size of the input. However, if T meets the similarity assumption (as described in Ahuja et al. (1993)), that is, if T is polynomially bound by a non-number input parameter, such as the size of the network, represented by $v = |V|$, where V is the set of nodes of the network, so that $T = O(v^k)$, for some k, then the algorithm will run in polynomial time.

(2008) formulate an evacuation problem they call the Building Evacuation Problem with Shared Information (BEPSI). BEPSI has the goal of finding the set of routes in time-varying flow networks such that the total time of all occupants to reach safety is minimised (the quickest flow problem). Additionally, through problem constraints, they consider the issue of shared information during a building evacuation, that is, that since routing plans must be communicated to individuals at decision points and communicating different plans to different individuals could introduce confusion they argue that groups of individuals should receive common routing instructions. The BEPSI problem is shown to be NP-hard and they propose an exact solution that has an exponential worst-case computational complexity.

Up until this point, the works covered here have assumed that network attribute values are known and deterministic; however, some works have also considered that during evacuation it is possible to face node or edge failures, such as when a passageway is obstructed or untenable. The possibility of failure is often modelled stochastically: for example, there may be some probability that an edge will become unavailable for usage at a particular point in time. We will now discuss works that include stochastic elements in their network models.

2.5.3 Stochastic Networks

Several researchers have addressed evacuation planning through models that include stochastic elements. Some have studied stochastic flow problems as network connectivity and reliability problems, where nodes or edges may fail randomly with known probability. While not applied directly to evacuation planning, these works are still relevant since they consider various problem objectives suited to stochastic environments. Others have applied queuing networks to evacuation problems, particularly for studying issues of congestion during egress. We first review works investigating stochastic flow problems then those concerning queuing networks.

The following works produce network connectivity or system reliability measures or metrics that are useful for evaluating network performance. The purpose of network connectivity analysis is to evaluate the probability of the connectedness of nodes, e.g. is there a path from the source node to the sink node. The goal of system reliability is to assess the probability that a network can accommodate a particular level of flow. A comprehensive review of the area of network reliability is given by Ball et al. (1995), and for an overview of the computational complexity of reliability problems, see Ball

(1986).

Mirchandani (1976) investigated network connectivity problems and problems of finding the expected travel time of paths in probabilistic networks where travel time is stochastic, and presented algorithms for both. Frank and Gaul (1982) looked at various kinds of connectedness probabilities. Due to the complexity of finding exact values, bounds and approximations to the probabilities are provided. In fact, the problem of evaluating probabilistic connectedness was later shown to be NP-hard (Ball, 1986). Due to the complexity of solving these problems exactly, works have suggested approximation methods, and central to these works are methods based on Monte Carlo sampling approaches, for example, Fishman (1986); Fishman and Shaw (1989).

More recently, in a series of works (Lin, 2001, 2002a,b, 2003), Lin considered system reliability in stochastic but time-invariant flow networks where nodes and edges can fail. In (Lin, 2001, 2002b), the goal is to evaluate the system reliability — the probability that the maximum flow of the network is not less than a given supply. Several algorithms are given for tackling this problem. The work is extended to the multi-commodity case in (Lin, 2002a). In (2003), Lin extended the Quickest Path Problem (restricted case of the Quickest Flow Problem previously discussed) to the system reliability evaluation in stochastic-flow networks.

Another approach for examining network performance of stochastic, but static, flow networks with edge or node failures is to consider the expected value of the maximum flow. This problem is, however, also known to be NP-hard because, in summary, it contains the probabilistic connectedness as a sub-class of problems (as discussed in (Nagamochi and Ibaraki, 1992)). Due to the general complexity, upper and lower bounds on the exact value have been proposed as approximations (Onaga, 1968; Carey and Hendrickson, 1984). Later, Nagamochi and Ibaraki (1991, 1992) detailed necessary and sufficient conditions for these bounds to equal the exact value.

For the analysis of the performance of evacuation systems more generally, Løvås (1995) provides an extensive discussion on evacuation performance measures and how to calculate them. Løvås presents a stochastic model of an evacuation system and discusses how queuing network models can in theory be used to calculate the performance measures, and demonstrates how in practice simulation methods can be used to calculate such measures.

Note that no routing plans are developed in these works. Furthermore, they do not consider the time-varying nature of network attributes that could be important in the modelling of emergency scenarios, as discussed previously.

Others have investigated stochastic, dynamic network models where routing plans are developed but network properties are time-invariant. Karbowicz and MacGregor Smith (1984) applied stochastic network models to evacuation problems in order to find optimal evacuation routes for building occupants. They propose a simulation-based heuristic approach based on the k-th Shortest Paths approach. Their method is expected to perform poorly in congested networks due to significant queuing. Talebi and MacGregor Smith (1985) compared analytical queuing network models and simulation for solving the stochastic evacuation problem. They used the expected total evacuation time as the performance measure, and concluded that queuing networks were effective for modelling their evacuation scenarios.

2.5.4 Opasanon's Safest Escape Problem

Of particular relevance to the work in this thesis is Opasanon's Safest Escape Problem (Opasanon, 2004; Opasanon and Miller-Hooks, 2008). Opasanon and Miller-Hooks model the uncertainty and transiency of emergency evacuation scenarios using network flow problems modelling network edge properties as time-varying and stochastic. Whereas previous flow problems for evacuation, in general, used either earliest arrival or maximum flow, or minimum travel time as the criteria by which to evaluate flow patterns, here the goal is to maximise the probability of successful traversal from source to sink. In other words, given that edges and hence paths have a probabilistic distribution of capacities, the goal is to find the paths from source to sink that have the highest likelihood that the required capacity will be available. In a sense, this problem objective is a hybrid of a reliability measure with a minimum cost flow problem.

Specifically, they formulated the Safest Escape problem (SE) with the goal of finding the *a priori* flow pattern where the probability of the path with the minimum probability of successfully traversing a network from source to sink is maximised[4]. Rather than focussing on a system-wide objective, e.g. minimum total time to exit, the SE provides evacuation plans such that the risk taken by occupants who take the greatest risk is minimised. Waiting is not permitted at nodes, and they focus on single-source, single-sink networks[5]. They provide an exact algorithm, the Safest Escape Algorithm (SEA), for solving the case where edge attributes are time-varying, and capacities are stochastic but traversal times are deterministic. The SEA is based on the Successive

[4]This is analogous to the problem of minimising last time of arrival versus the total travel time.

[5]Networks with multiple sources and multiple sinks can be converted using the efficient technique proposed in (Miller-Hooks and Patterson, 2004).

Shortest Path Algorithm (see (Ahuja et al., 1993) for details) and has pseudopolynomial time complexity.

2.5.5 Stochastic, Time-Varying Flow Networks

In his thesis (2004), Opasanon goes on to discuss flow problems in networks where both travel times and capacities are time-varying and stochastic (herein shortened to STV Networks). The rationale behind adopting this model is that in reality both attributes are likely to be uncertain. Certainly, in the previously discussed context of future response systems, where, for example, models of smoke spread may provide predictions about future building smoke conditions, it is possible that different likelihoods will be attached to the forecasts.

Nevertheless, solving flow problems in STV Networks is much harder than in the network models considered previously. Underlying many efficient solution approaches to simpler network flow problems are efficient shortest path (SP) techniques (e.g. the Successive Shortest Path Algorithm) that rely on Bellman's Principle of Optimality (PoO) (Bellman, 1957). PoO enables flow algorithms (through subroutine shortest path techniques) to efficiently exploit the underlying graph structure to find optimal solutions. However, in STV Networks the PoO no longer applies, as for example shown by Hall (1986) for the minimum expected travel time problem in uncapacitated[6] networks with stochastic, time-varying travel times. Furthermore, Pretolani (2000) showed that the minimum expected time problem for simple paths[7] in uncapacitated STV Networks is NP-hard.

It is perhaps due to this complexity that most work in STV Networks has focussed on finding optimal paths for both *a priori* and *adaptive* policy problems in uncapacitated STV Networks (see (Gao and Chabini, 2006) for a comprehensive review). A priori methods assume a fixed path must be chosen before any information is realised so that the result is a single path, where for each node in the path, say i, the path would include a single subsequent node j, no matter the actual arrival time at node i. These problems are equivalent in formulation to single-stage stochastic programming problems — static formulations with no recourse. On the other hand, adaptive policies allow for time-adaptive route choice, that is, the result is not a single path but an adaptive policy, so that, for example, the subsequent node j from a node i depends on the actual arrival time at i. Adaptive policies are equivalent to multi-stage stochastic pro-

[6]Networks in which edges do not have capacities but have, for example, travel times.

[7]Simple paths contain no repetitions of nodes (cycles).

gramming problems that allow recourse at each stage[8]. Similar to (Opasanon, 2004; Opasanon and Miller-Hooks, 2008), the focus of this thesis in on a priori problems.

To the knowledge of the author, no exact methods, efficient or inefficient, have been proposed for solving flow problems in STV Networks. However, several heuristic approaches have been developed. Opasanon (2004) and Opasanon and Miller-Hooks (2010) developed an Evolutionary Algorithm for the a priori minimum expected time flow problem in STV Networks and also discussed extensions to multiobjective problems. Later Miller-Hooks and Sorrel (2008) extended the same approach and applied it to the problem of finding the maximal expected flow in STV Networks. Details of these works are provided in Sections 3.5.2.1 and 3.5.2.2, respectively, in order to appropriately situate them in the literature on Evolutionary Algorithms in stochastic combinational optimisation (Section 3.5).

2.5.6 Conclusions

The sections on network optimisation for evacuation planning have provided a survey of works investigating a number of network models applied to evacuation problems. They have illustrated the wide array of network models and flow problems in the literature. They have also highlighted that the incorporation of more complex network elements, such as time-varying and stochastic network attributes, while arguably increasing the realism of the models, substantially increases the computational complexity of solving optimisation problems. Due to this increasing complexity, novel approaches based on approximation and metaheuristic methods are beginning to be applied.

2.6 Summary

The first chapter of the literature review has covered a number of works related to the planning of emergency evacuation. It began with a brief discussion of online decision-support systems for emergency evacuation, mentioning several works that expound the requirements for such systems. In particular, highlighting a recent project, FireGrid, that proposed a future emergency response system for the built environment with the goal of providing the necessary infrastructure to allow the harnessing up-to-date information about unfolding incidents combined with analytical tools to enable better

[8]See (Birge and Louveaux, 1997), for example, for more details on stochastic programming formulations.

informed response decision-making. It is in the context of this project that the work described herein is situated.

The review then briefly discussed simulation models used for evacuation planning, distinguishing between micro- and macroscopic models. Microscopic models work at a high-level of granularity representing occupants heterogeneously and hence focus on the differences between people in an attempt to predict how occupants might evacuate from a building. On the other hand, macroscopic models represent evacuation at a low level of granularity such that people are modelled as homogeneous entities, and the goal is to provide a lower bound on, for example, the duration of evacuation. The focus of this thesis is on macroscopic models, in particular flow network models. Additionally, comments were made about time as the principle decision parameter in evaluating evacuation, and that in the context of future emergency response systems using real-time information about the environment, new dimensions for evaluating risk may be presented. This is followed by a brief discussion on the modelling of emergency evacuation using flow networks, explaining in general the mapping between building circulation systems and the components of network flow models.

The rest of the chapter reviewed works addressing evacuation problems in flow network models. A number of key flow problems are highlighted in relation to evacuation, and a number of network models are surveyed from dynamic to time-varying models, to those including stochastic elements. Particular focus is given to several works addressing flow problems in network models that include elements that are stochastic and time-varying.

Overall, this chapter reviewed works that situate and motivate the flow problem addressed in this thesis. The following chapter focusses on works related to the proposed solution approach.

Chapter 3

Literature Review: Evolutionary Algorithms

In this chapter, we give an overview of the framework of the solution approach proposed in this thesis, Evolutionary Algorithms (EAs). It begins with a general introduction to EAs and common EA variants, and then details how they have been applied to relevant deterministic combinatorial optimisation problems[1] (COPs). Next it discusses their use in noisy and stochastic environments, and in particular their application to stochastic combinatorial optimisation problems (SCOPs).

3.1 Introduction

Evolutionary Algorithms provide general algorithmic frameworks for tackling global optimisation problems, and, as such, form a subclass of so-called metaheuristic algorithms (Gendreau and Potvin, 2010). EAs are distinguished from other metaheuristics in that they are inspired by the principles of Darwinian evolution[2]. In particular, through the metaphorical adoption of evolution they harness the principle of the survival-of-the-fittest. This powerful idea has led to the successful application of EAs to a number of COPs, and more widely (see, for example, (Bäck et al., 1997) for a brief

[1]COPs are discrete optimisation problems where the solution space is defined over combinatorial spaces, such as graphs, matroids, etc. The classic goal of COPs is to maximise or minimise a real-valued objective function on a large but finite set of feasible solutions. Typical examples include Travelling Salesman, Vehicle Routing, Integer Programming, Minimum Spanning Tree and Knapsack problems. Deterministic COPs are problems where all problem data is known a priori and with certainty. For an introduction to COPs, see, for example, Papadimitriou and Steiglitz (1982).

[2]Compared with, say, swarm-intelligence inspired algorithms such as Ant Colony Optimisation and Particle Swarm Optimisation, or Simulated Annealing and Tabu Search.

list of applications).

Many EA variants exist; however the majority of approaches are based on three independently developed frameworks: Genetic Algorithms (GA) invented by Holland (1975); Evolutionary Programming (EP) proposed by Fogel et al. (1966); and Evolutionary Strategies (ES) proposed by Rechenberg (1973). Note, that, generally, EP and ES are applied to continuous optimisation problems, and because the focus of this thesis is on discrete optimisation problems, much of the following discussion will be limited to GA-type approaches. For general references to GAs, see, for example, (Goldberg, 1989; Mitchell, 1998). For a comprehensive introduction to EAs, see (Bäck et al., 1997). Recent overviews of EAs include (Calégari et al., 1999) and (Hertz and Kobler, 2000), and more generally, overviews of metaheuristics, including EAs, applied to COPs are (Blum and Roli, 2003) and (Gendreau and Potvin, 2005), and to SCOPs (Bianchi et al., 2006b).

EAs, in general, are regarded as robust, heuristic, “black-box” methods, that is, heuristic methods that put few requirements on the search space, e.g. they do not demand convexity or continuity[3], and hence are particularly apt for solving hard COPs with large search spaces, where the spaces are, for example, nonconvex[4] or simply uncharacterised (Bäck et al., 1997). EAs are often applied to NP-hard combinatorial optimisation problems where instances sizes are beyond the reach of exact methods that have exponential computational complexity, because, in general, they cannot compete in time or accuracy on problems that permit efficient exact techniques. Additionally, EAs are well-suited to multi-objective optimisation problems and are easily parallelisable. These properties make them attractive to practitioners who are faced with complex, often highly constrained, real-world problems. However, EAs offer no panacea, since, while easily applicable to many problems, there are no guarantees that they will necessarily find optimal solutions, or even solutions within some predetermined distance from the optimal. Thus they are primarily suitable for scenarios that allow a trade-off between the reliability of the method and its computational cost, or where no efficient exact method is likely to exist, e.g. NP-hard problems.

We now give an overview of the general EA framework, then describe common algorithmic components and the overall functioning of the approach.

[3]Convexity (concavity) implies that a local minimum (maximum) is also a global minimum (maximum), and continuity implies that the objective function is continuous over, for example, the real numbers, $\mathbb{R}$, so that small changes in the input reflect small changes in the output.

[4]COPs that have nonconvex search spaces may have a number of local and global extrema.

3.2 Algorithms

Similar to other metaheuristics, EAs work in an iterative fashion, attempting to improve on the best solution discovered while also exploring the problem's search space, until some termination criteria are met. However, unlike some metaheuristics, such as Simulated Annealing and Tabu Search, they are population-based approaches, since a pool, or "population", of candidate solutions is maintained throughout the search. The "population" acts as a memory of points searched in the space, and, in fact, EAs proceed in a Markovian fashion since the information used at any point in time is exclusively based on the current state of the search.

Algorithm 3.2.1: GENERIC EVOLUTIONARY ALGORITHM(μ, λ, p_c, p_m)

```
$i \leftarrow 0$
$\mathcal{P}_i \leftarrow$ InitialisePopulation($\mu$)
Evaluate($\mathcal{P}_i$)
while (termination conditions not met)
  do {
    $i \leftarrow i + 1$
    $\bar{\mathcal{P}} \leftarrow$ SelectMates($\mathcal{P}_{i-1}, \lambda$)
    $\mathcal{O} \leftarrow$ Crossover($\bar{\mathcal{P}}, p_c$)
    Mutate($\mathcal{O}, p_m$)
    Evaluate($\mathcal{O}$)
    $\mathcal{P}_i \leftarrow$ SelectReplacements($\mathcal{P}_{i-1} \cup \mathcal{O}, \mu$)
```

The generic EA presented in Algorithm 3.2.1 from Blum and Roli (2003) proceeds in the following way. First an initial population of size μ is generated and each member is evaluated, where each member of this "population" represents a potential solution to the problem at hand. The algorithm then enters an iterative process that continues until some termination criteria are met. During each iteration, or "generation", a number of operators are applied to the current population of candidate solutions. A "mating" selection operator is applied to select pairs of parents for "breeding", during which a "crossover", or recombination, operator is used to generate a number, λ, of new individuals from the mating population to create a set of "offspring". Next a "mutation" operator is applied to the offspring to introduce "genetic variations" to the population and hence increase the diversity of individuals. The rate at which the genetic operators, crossover and mutation, are applied is controlled by input parameters, p_c and p_m, respectively. The progeny are then evaluated against a problem-specific fitness function and each is assigned a fitness value, that is, a measure of the quality of the solution represented by the individual for the given problem, or “environment”. This value, implicitly or explicitly, dictates the likelihood that each solution will “survive” to the succeeding generation (iteration). Finally, a replacement strategy and selection operator are combined to choose μ individuals from the current population and offspring based on their fitness values to constitute the population of the next iteration.

The basic components of an EA (the functions in Algorithm 3.2.1) as well as a solution representation must be properly defined in order to specify a functioning EA. The required elements are:

- Solution Representation: the encoding used to represent candidate solutions, and hence individuals in the population.
- Initialisation: method by which the initial population is generated.
- Mating Selection: operator that selects parents with which to generate new individuals.
- Crossover: operator used to generate new individuals based on pairs of parent individuals.
- Mutation: operator used to perturb new individuals in order to introduce genetic variations into the population.
- Evaluation: method that assigns a quantitative measure of solution quality for a specific problem.
- Replacement Strategy: strategy by which the members of the current population are replaced each iteration.

It is also necessary to define termination criteria that determine when the search process terminates; and furthermore, it may also be required to address issues relating to problem constraint handling and candidate solution feasibility. The choices made for specific operators and strategies are not independent, and will depend on various factors, perhaps the most important of which are the problem being tackled and the choice of solution representation. We now discuss typical EA elements, drawing heavily on (Bäck et al., 1997).

3.2.1 Solution Representation

One of the first and most important decisions for the application of EAs to COPs is that of a representation for solutions. The particular representation will depend on the specifics of the problem at hand, and, furthermore, there may in fact be various ways in which to represent candidate solutions. Notwithstanding the underlying representation, and by analogy to genetics, a solution is called a *chromosome*, with each chromosome consisting of a number of *genes*, and hence a chromosome corresponds to the *genotype* of a candidate solution. Each gene can adopt a number of different values, or *alleles*, and the position of a gene in the chromosome is referred to as the *locus* of the gene. Often each locus will have a distinct interpretation in terms of the problem and the solution's *phenotype* (i.e. the 'actual' solution which the chromosome represents). For the canonical GA, for example, chromosomes are typically represented by binary strings, where each bit denotes a single gene and, hence, can at any time adopt a single value drawn from $\{0,1\}$. Other classic encodings include Gray codings, real-valued vectors (ordered lists), and permutation encodings. See (Bäck et al., 1997) for more details on typical representations.

An additional issue with the representation of solutions for COPs concerns the feasibility of candidate solutions with respect to problem constraints. Problem formulations may include constraints on the values decision variables[5] can adopt, and it is often required to consider these on top of problem objectives. Some approaches for constraint handling involve modifications to solution representations, which can often increase the general complexity of the algorithm, requiring further changes to operators such as crossover. General constraint methodologies are often developed to account for the specific constraints introduced by the use of a specific formulation. This is discussed more in Section 3.2.9.

3.2.2 Initialisation

The first step of an EA is to generate an initial population (of size defined by the parameter μ). The initial population acts as a starting point for the search and, hence, this set of solutions is important for the success of the search. Situations may arise where little is known about the search space and a typical approach in these cases is to randomly sample from the search space, so that all candidate solutions that can be

[5]The decision variables are the variables that the decision maker is in control of, in, for example, an optimisation problem.

represented are equally likely to be included in the initial set. Another possibility in order to increase the rate of convergence is to seed the initial population with solutions discovered by, for example, heuristics or greedy methods, that will generally introduce a bias into the direction of the search in order to focus on particular areas of the search space. Due to the often (pseudo-)randomisation of the initial population, this acts as the first stochastic element in the run of an EA and, therefore, it is common to run an EA several times in order to get a sense of its average performance over different initialisations.

The size of the population, denoted by μ, is an important consideration here. A population size that is too small will potentially under-represent the solution space and could lead to quick, sub-optimal convergence to a particular area of the search space. In practice, the convergence of the population involves converge to the same genotype or to the same fitness value. Conversely, too large a population would require unnecessary additional computational effort for little gain. The optimal population size is generally problem-specific and dependent on the complexity of the problem and its solution space, the chosen selection operators and other parameters settings.

For COPs, a further issue arises in relation to the feasibility of the initial solutions, specifically, about whether it is necessary, or even possible, to generate an initial population where all solutions are feasible given the problem constraints. For the types of problems to which EAs are often applied it is in itself a hard problem to generate feasible solutions (Coello Coello, 2002). This is part of the wider issue of dealing with problem constraints in EAs.

3.2.3 Mating Selection

The first operator to be applied each iteration is the *mating selection* operator, which selects λ parents that will be used to generate λ offspring. The crossover operator (next section) determines how the children will be generated. Typical selection mechanisms include fitness proportionate selection, tournament selection, rank-based selection and random selection. We now give an overview of these selection schemes; for more details, see (Bäck et al., 1997).

Fitness proportionate selection For this selection operator, individuals are assigned a selection probability that is proportional to the fitness of the individual given the current population. For selecting the individuals, a sampling mechanism called the *roulette wheel* sampling algorithm is used to select the mating population. In this method, one can think of the probability distribution as defining a roulette wheel on which each slice has a width corresponding to the individuals selection probability. Sampling from the distribution is then like spinning the roulette wheel and testing which slice ends up at the top. An individual is selected after each spin of the wheel, and therefore to select λ parents requires λ spins of the wheel. Baker (1987) introduced a method for sampling individuals with lower variance, called the stochastic universal sampling (SUS) algorithm, which is commonly used instead of the roulette while approach. This method is efficient requiring only a single pass over all the members of the population to select the mating population.

Tournament selection For this scheme, individuals are randomly chosen with or without replacement from the current population to compete in a tournament for selection. Commonly, binary tournament selection is utilised, whereby two individuals are chosen to compete and the individual with the higher fitness wins and is selected for mating. The determination of a tournament winner may be deterministic, or stochastic so that the worse individual has some probability of being selected. The tournament size, the number of individuals involved in each tournament, can be generalised to an n-Tournament selection operator.

Rank-based selection In this scheme, individuals are allocated a selection probability based on their rank ordering in the current population. This is in contrast to fitness proportionate schemes, since the selection probability is based solely on the rank of the individuals not their fitness values. A number of rank-based operators exist including linear and non-linear operators. Once selection probabilities are determined, the SUS algorithm mentioned previously can be used to select the mating population.

Generally, rank-based operators are considered more robust than fitness proportionate selection mechanisms (Whitley, 1989).

Random selection Unlike the previous strategies, this operator, often used with ESs, chooses parents without direct consideration of their fitness values, that is, parents are chosen uniformly randomly with replacement from the current population.

3.2.4 Crossover

Each iteration new individuals are generated using a crossover operator, which combines the genes of a pair of selected parent individuals from the current population. Commonly, a pair of individuals (parents), chosen by the mating selection operator, are recombined to generate two new individuals (offspring). The particular implementation of the operator depends on the solution representation and also on other issues such as the need to preserve feasibility. As an example, for binary string representations, one- or two-point crossover and uniform crossover are commonly used. We now briefly describe these operators; again, for more details, see Bäck et al. (1997).

One-Point Crossover For this operator, a single crossover point is selected for both parents, and the genes beyond the chosen point are swapped between the parents to generate two offspring. For example, for two parents using binary representations, where parent $P_1 = 11111$ and $P_2 = 00000$, say a crossover point 2 is selected, then the resulting offspring are $O_1 = 11000$ and $O_2 = 00111$.

Two-Point Crossover Similar to the above operator, except that two crossover points are selected and the genes between the two points are swapped to again generate two offspring. Using the previous parents, $P_1 = 11111$ and $P_2 = 00000$, and two crossover points, say 2 and 4, then the offspring are $O_1 = 11001$ and $O_2 = 00110$.

Uniform Crossover In contrast to the previous operators, this operator considers each locus for crossover and, given some probability, swaps the parents' genes value at each locus to produce two offspring.

For more complex representations, such as ordered lists, more elaborate operators may be required, particularly in relation to feasibility.

Typically, crossover has a parameter, $p_c \in [0,1]$, that determines the rate at which the crossover operator is applied to pairs of parents. In the case that it is not applied, then the parents pass unaltered to the offspring population. Typical values for $p_c \in [0.5, 0.95]$ (Bäck et al., 1997).

3.2.5 Mutation

The next operator to be applied is the mutation operator. The purpose of mutation is to perturb the newly generated offspring so as to introduce genetic variations (i.e., genes that could not be produced by the crossover of the parents) into the population.

Mutation is considered to serve as a ‘background operator’ to crossover; however, its function is still critical, and in terms of search it has the important role of introducing ‘noise’ into the population in order facilitate the escape from local optima.

As is the case for crossover, different standard operators are available (Bäck et al., 1997). However, a typical operator will simply replace a gene with another possible allele. For example, for a binary string representation, a gene can be mutated by inverting its value, i.e. a gene with value 1 is flipped to a 0. Again, solution feasibility may have to be considered depending on the mechanics of the operator and the constraint handling methodology.

Typically, a mutation rate, $p_m \in [0,1]$, is specified as a parameter that determines the likelihood that any given gene of an offspring individual is mutated. Common values are $p_m \in [0.001, 0.1]$ or even $1/l$, where l is the number of genes in a chromosome (Bäck et al., 1997).

3.2.6 Evaluation

The purpose of the evaluation is to assign a quantitative measure of the quality of a candidate solution to a specific problem. The quality is termed the ‘fitness’ of the individual and it is assigned using a fitness function $f : \Omega \rightarrow \mathbb{R}$, where Ω is the search space. The definition of the fitness function is problem dependent, and for COPs it is often defined directly as the problem objective of the mathematical programming formulation, when this is available. However, fitness functions vary enormously depending on the problem; for example, they could be defined to evaluate a simple closed-form expression, or run and appraise large-scale simulations, or even call on valuation by human users. The variation in fitness functions means that often evaluating the fitness of individuals, and hence executing the evaluation step, is the most computationally demanding stage of an EA’s generation.

An additional aspect of the evaluation of individuals involves the use of decoders. A candidate solution is encoded in the form of some representation and often a decoder is necessary to transform this chromosome to give it a meaning with respect to the actual problem. For example, before it can be evaluated it might be necessary to decode a chromosome represented as a binary string to the list of integers it represents, where each integer is represented by a number of genes (bits).

A number of issues may arise with the choice of the fitness function for a particular problem. It could be that the computational requirements of the function make it expensive to evaluate many candidates; for example, if the evaluation requires the execution of a large-scale simulation. Or it may be possible that the fitness function is noisy, due, perhaps, to the use of randomised simulations for evaluating candidates, and therefore it may not be possible to determine the true fitness of individuals. The issue of uncertain environments, in particular stochastic and noisy environments, is dealt with extensively in Section 3.4.

For COPs, once again, the feasibility of solutions may need to be considered by the fitness function. In fact, several of the popular methods for handling constraints, for example penalty functions, rely on the modification of the fitness function to cope with constraint violations. For more details, see, for example, Coello Coello (2002).

3.2.7 Replacement Strategy

Given the iterative nature of EAs, replacement strategies define how each succeeding population will be generated from the current population plus its offspring. A common notation originally from the ES literature is used to denote the replacement strategy in terms of the population size, μ, and offspring size, λ. Two general strategies have been identified: the *Comma*, denoted as (μ,λ), and the *Plus*, $(\mu+\lambda)$, strategies. The Comma strategy is that at each generation the new population is replaced by the best μ individuals from the offspring population of size λ, where $\lambda \geq \mu$; whereas for the Plus strategy the succeeding population of μ individuals is selected from the joint collection of the current population and the offspring.

Typical replacement strategies include:

- Generational replacement: $(\mu,\lambda=\mu)$; the current population is completely replaced by μ offspring. This strategy is often combined with elitism, where the best individual/chromosome from the current population gets a free pass to the next generation and hence $\mu-1$ individuals are selected from the offspring to form the new population. Elitism is employed to prevent to the loss of the currently best individual due to stochastic mating selection operators.

- Comma-replacement: (μ,λ); a more general replacement strategy whereby the current population is replaced by μ individuals from the offspring, where $\lambda \geq \mu$.

- Steady-state replacement: $(\mu+1)$; the next population of size μ is selected from the current population plus a single offspring.

- Plus-replacement: $(\mu + \lambda)$, where $1 \leq \mu \leq \lambda < \infty$; this strategy implements a deterministic elitist replacement strategy that is a generalisation of the steady-state scheme, whereby the best μ from current population and the offspring, a total pool of $\mu + \lambda$ elements, replace the current population.

A comprehensive list of replacement strategies can be found in Bäck et al. (1997).

Note that the choice of replacement strategy should be balanced with the choice of mating selection operator to ensure that an appropriate level of selection pressure[6] is applied.

3.2.8 Termination

Finally, termination criteria are also required. These are the criteria that determine when the algorithm terminates the search. Common criteria include a maximum number of iterations (or generations) after which the individual with the highest fitness is returned as the solution; or possibly convergence criteria so that the process terminates should all the members of the population converge to a single chromosome. A further possibility is to terminate should no progress be made, that is, for example, if the fitness of best found individual does not improve over a number of generations.

3.2.9 Constraint Handling

An important, additional issue with the application of EAs to COPs is the handling of problem constraints. Many techniques have been developed to handle constraints in EAs. Sometimes problem specific techniques are developed, but many general approaches have also been developed, such as the use of penalty functions, repair algorithms, decoders, and constraint-preserving operators. For an extensive survey, see Coello Coello (2002).

A core decision is whether to allow infeasible solutions into the population or not. If they are allowed, then it is necessary to devise a method by which infeasible solutions are penalised. A common approach is to utilise penalty functions that add additional terms to the fitness function to reflect the degree of constraint violation. If infeasible solutions are prohibited within the population, then it is necessary to guarantee that, firstly, the initial population is feasible, and secondly, that the genetic operators

[6]Selection pressure is a parameter that characterises EA selection operators (Bäck et al., 1997). Essentially, selection pressure is the degree to which the better individuals of a population are favoured during selection. The higher (lower) the selection pressure, the more (less) better individuals are favoured.

guarantee the feasibility of any new individuals that are generated. This may involve, for example, repair algorithms that repair infeasible solutions so that they are feasible given problem constraints, or constraint-preserving operators.

3.3 EAs in Combinatorial Optimisation

Next, we discuss the application of EAs to related COPs, namely shortest path and network flow problems. This is followed by sections reviewing works investigating the use of EAs in noisy environments, and, finally, applications of EAs to SCOPs.

3.3.1 EAs for Shortest Path Problems

The application of EAs to network flow problems, and even shortest path problems (SPP), is not common. This is most likely to do with the availability of efficient, exact algorithms for solving many flow problems in a variety of network models (as described in Section 2.5). Nevertheless, with the increase in the complexity of network representations that is evident from the literature, comes a significant increase in the computational complexity of flow problems, as seen, for example, with flow problems in Stochastic Networks, and in particular STV Networks.

Here we first discuss applications of EAs to shortest path problems and then, secondly, to problems in flow networks.

Gen et al. (1997) applied GAs for solving the SPP. They recognised the difficulty in encoding a path in a graph, that is, paths can have variable number of nodes (and edges), and a random sequence of edges does not necessarily correspond to a graph. To handle this they developed a priority-based representation, which is related to the permutation representation, for encoding the paths of a graph. In this encoding, all nodes are represented by genes and the associated value of each gene is used as the priority of the node for constructing a path among candidates, so that for a node 1 with neighbours $\{2,3,4\}$ and gene values $\{3,6,7\}$, respectively, the priority value determines that node 1 is connected to node 4 in the path represented by the chromosome. However, it is possible that different chromosomes, different sets of priorities, may result in the same path. For genetic operators, they use modified operators used for permutation representations. To evaluate the approach, the GA is compared with optimal solutions for a set of test problems. Overall, the GA found optimal solutions with a high frequency, although the comparative computational cost of the GA is not commented on.

Later work (Gen et al., 2001) provides a summary of the application of GAs to various network design problems and the bi-criterion SPP. The earlier approach is adapted to handle bi-criterion problem objectives.

Several other works have applied GAs to the SPP (Inagaki et al., 1999; Ahn and Ramakrishna, 2002). Inagaki et al. (1999) proposed a GA using fixed-length chromosomes to encode paths. Chromosomes are sequences of integers and each gene represents a node ID that is selected randomly from the adjacent nodes corresponding to its position number in the route. A simple crossover operator is used in order to maintain feasibility of routes. Ahn and Ramakrishna (2002) also propose a GA for solving the SP problem. Their genetic representation is simpler and allows for variable-sized chromosomes. Each chromosome is a path where each gene encodes a node in the path. The position of the node in the path is given by the position of the gene. They propose a crossover mechanism that requires parent chromosomes to share a common node; however, this method can introduce cycles to the path, which make them infeasible given the problem assumptions. A simple repair function is utilised to remove the cycles from paths. The algorithm is experimentally evaluated and shown to perform better than the algorithm in Inagaki et al. (1999).

A few works have also proposed GAs for solving stochastic shortest path problems (SSPP) (Ji, 2005; Davies and Lingras, 2003). Ji (2005) considers the SSPP and several variants, and proposes a hybrid solution approach, combining genetic algorithms with stochastic simulation. Similar to Ahn and Ramakrishna (2002), a variable-length chromosome representation of a path is used, and genetic operators are defined identically as well. Davies and Lingras (2003) apply a GA approach to the dynamic SSPP, that is, where edge weights change as unknown functions of time. In their approach the GA is used to reroute in order to find shortest paths given the changing network information. The algorithm is split into two components: a Prediction Module and a GA. The Prediction Module is used to provide updated edge travel times and the GA for finding the shortest path given the up-to-date environment information. Like Ahn and Ramakrishna (2002), a chromosome is defined as a path (although it can be cyclic), where each gene is a node in the path, and, hence, it is also of a variable length. Given the dynamic nature of the network, they also allow waiting in paths. Similarly, a simple crossover operator is defined whereby parents can only recombine if they contain shared nodes, which, again, maintains the feasibility of offspring.

3.3.2 EAs for Network Flow Problems

Hitherto, discussed works have only considered uncapacitated networks; in fact, only a few works have proposed EAs for flow problems in capacitated networks. Munakata and Hashier (1993) applied GAs to the maximum flow problem in static flow networks. Each candidate solution is represented by a flow matrix, and a complicated scheme is required to handle the flow constraints. Because of this complexity, special genetic operators are designed which do not however guarantee feasibility of candidate solutions, and, hence, a penalty procedure is used to decrease fitness values. Experimental results showed that the GA performed badly in comparison with exact procedures.

Several works have applied EAs to related flow problems in traffic assignment[7]. An example is (Sadek et al., 1997), who used GAs for addressing dynamic traffic assignment problems, with the goal of minimising the total travel time that vehicles spend en route. In their model, traffic flow limitations are imposed through capacity constraints. Chromosomes are represented by real-valued vectors (ordered lists), where each vector element is the number of vehicles assigned to an edge during a specific time period. Constraints are addressed through the use of a penalty function. The GA is compared against a non-linear programming method and shown to be a viable alternative approach, with several advantages: one, that by using it some of the assumptions required for the analytical approach can be relaxed, and, two, that it can handle larger instances. In (Varia and Dhingra, 2004), a GA is applied to the dynamic optimal traffic assignment problem with the dual goals of minimising the overall travel cost (actually travel time) in time-varying networks and the optimisation of signal timings. For representation, a binary-string chromosome is utilised that encodes each source-sink path and the level of flow assigned to it, and additional bits are added to represent signal timings. A penalty function is also used to punish individuals for signal timing constraint violations. The GA is evaluated on example networks and found to handle the problem's complexity well. Varia and Dhingra also conclude that one of the main advantages of GAs is that assumptions can be relaxed that may be required for analytical approaches, e.g. convexity.

As previously mentioned, of particular interest here are several flow problems addressed by Opasanon (2004), Opasanon and Miller-Hooks (2010), and Miller-Hooks and Sorrel (2008). However, again, discussion of these works is delayed until Section 3.5 where they will be described along with other EA approaches to SCOPs in order

[7]Traffic assignment problems concern the selection of routes (or paths) between origins and destinations in transportation networks.

to situate them more widely in context.

3.3.3 Conclusions

In summary, as previously mentioned, few works have applied EAs to network flow and closely related COPs. The main conclusion from this section is, as might be expected, that when efficient approaches are available EAs cannot compete in either accuracy or efficiency. Nevertheless, EAs can be useful when larger, perhaps more realistic, problem sizes are to be tackled.

We now discuss at some length the use of EAs in noisy environments, followed by a review of EAs in stochastic combinatorial optimisation.

3.4 EAs in Noisy Environments

The particular problem addressed in this thesis, the Maximal Safest Escape problem (formally defined in Section 4.3), is formulated as a SCOP. Therefore, previous work tackling SCOPs and problems in noisy environments more generally are relevant here. We begin with an overview of the effect noise can have on the performance of EAs; then we survey theoretical and practical work on methodologies for coping with noise. Finally, we discuss applications of EAs to SCOPs, discussing in detail several highly relevant works.

Jin and Branke (2005) survey works investigating EAs applied to optimisation problems containing a wide range of uncertainties. In particular, they identified four types of uncertainty:

1. Noise: due to the presence of noise one is unable to measure the exact quality of an individual, e.g. variations and noise from sensor measurements or randomised simulations.

2. Robustness: optimal solution should be robust to perturbations, or slight changes, in design variables.

3. Approximated Fitness Functions: due to costly fitness evaluations, see (Jin, 2005) for a comprehensive survey.

4. Dynamic Environments: time-varying, deterministic fitness functions, where optimal fitness values are a function of time, and hence ‘dynamic’, see (Branke, 2002) for more details.

Note that evaluating solutions in stochastic environments using randomised simulation can be seen as equivalent to evaluating solutions in noisy environments, where fitness evaluations are subject to noise (stochastic variation) and hence modelled as random variables. The formulation below unifies this equivalency, and it is due to this equivalence that the survey focusses on EA approaches for handling fitness functions in noisy environments.

3.4.1 Impact of Noise on EAs

While known for being robust, EAs were originally designed for use in deterministic environments where fitness evaluations are assumed deterministic and tractable. However, the application of EAs to stochastic environments is not straightforward. In these environments, often the fitness evaluations are approximate and hence it is no longer possible to compute with finite resources the exact fitness evaluation of individuals. This, in turn, means that it is no longer possible to discriminate between individuals with certainty, thus inducing potential selection errors in the EA — that is, inferior individuals may survive and reproduce while superior individuals are eliminated. As a consequence the algorithm may be more likely not return the best solution discovered during the run. Ultimately, these issues can result in the algorithm being 'misled' and thus diminishing the effectiveness of the approach.

In general, in genetic search, there are two important issues in the evolution process of the search: exploration and exploitation. Exploration is the creation of population diversity by exploring the search space, and exploitation is the reduction of the diversity by focusing on the individuals of higher fitness or exploiting the information represented within the population. For effective search, a careful balance must be struck between the two. However, in noisy environments, where fitness values are uncertain, exploitation may be limited. To counter this it is possible to spend a potentially vast amount of effort accurately estimating the fitness of individuals. The question then arises of how precise estimates must be to facilitate effective exploitation, and whether the effort should instead be spent on, for example, more iterations of the algorithm.

Fitness functions that return fitness values that are noisy are called noisy fitness functions (NFFs), and EAs that utilise NFFs are called Noisy EAs (NEAs).

3.4.2 Noisy Fitness Functions

Numerous works address the use of NFFs, analyse the theoretical impact of noise on the performance of EAs, and also suggest a number of approaches for dealing with the noise.

Jin and Branke (2005) propose the following problem formulation for fitness evaluations under noise. Define $x \in X$ as the input configuration or design variables and the true fitness function $f : X \rightarrow \mathbb{R}$. Then the problem can be defined as follows.

Let the expected fitness function $F(x)$ be defined:

$$F(x) = \int_{-\infty}^{\infty} [f(x) + \delta] P(\delta) d\delta = \mathbb{E}[f(x) + \delta] = f(x), \delta \sim \mathcal{N}(0, \sigma^2) \tag{3.1}$$

where δ is additive noise and normally distributed with mean 0 and variance σ^2, and $P(\delta)$ is the density function of δ. Often, $F(x)$ is unavailable or too complex to evaluate in the closed-form, but the stochastic value $f(x) + \delta$ is available, so one can approximate the value by averaging the sum of a number of random samples:

$$\hat{F}(x) = \frac{1}{n} \sum_{i=1}^{n} [f(x) + \delta_i]$$

where n is the sample size and $\hat{F}(x)$ is an estimate of $F(x) = f(x)$. Justified by the law of large numbers, we know that as $n \rightarrow \infty$, the value converges on the actual fitness value. This is equivalent to, for example, the problem of approximating the expectation of a solution for an SCOP via simulation (e.g. using Monte Carlo methods), where $f(x)$ is equal to the expected value of a particular solution.

The most straightforward approach to reducing the effect of noise as an approximation to the true fitness value is to resample the fitness function a number of times and take the average of the fitness values. The main issue with this approach is that improving the estimation by increasing the sample size converges at a slow rate $O(1/\sqrt{n})$, and, furthermore, assuming that fitness evaluation is the most laborious part of an EA, increasing sample sizes to improve estimation may be costly.

Another issue with this approach is that an a priori fixed resampling rate does not consider possibly differing levels of noise across the space, such that δ may be dependent on x, $\delta(x)$. In this case, using too few samples means the estimates may provide poor fitness estimates and lead to incorrect selection; on the contrary, over-sampling could lead to the unnecessary use of computational resources. Due to the issues of choosing an appropriate sample size, many of the following works focus on methods that do not require increasing the sample size, n.

Jin and Branke (2005) split approaches to tackling noisy fitness evaluations (expressed by Equation 3.1) into four categories:

1. Explicit Averaging over time: the most common approach, using a fixed resampling rate[8] or adaptive sampling plan to estimate fitness, using the average of a number of fitness samples.

2. Explicit Averaging over space: base average fitness value on that of similar solutions, for example, by exploiting the neighbourhood of solutions;

3. Implicit Averaging: use a single fitness evaluation and increase the population size whereby the self-averaging nature of EAs will reduce the effect of the noise;

4. Modifying EA selection to account for the presence of noise.

Early works investigating the effect of noise on EAs suggested that EAs are quite robust to noise. Fitzpatrick and Grefenstette (1988) investigated the question of whether it is beneficial to spend more effort obtaining accurate estimated fitness values, or to accept less accurate estimations and use a larger population, thereby allowing a larger sampling of points in the space, given a fixed amount of computational effort. They concluded that in some cases increasing the population size was better.

Goldberg et al. (1992) proposed population sizing equations[9] to prevent the convergence of EAs being effected by the presence of noise. They also advocate the use of larger populations to deal with noise. Building on this work, Miller and Goldberg (1996); Miller (1997) developed theoretical models for determining the effects of noise on the operation of EAs. They proposed methods that simultaneously optimise the population size and the sample size for a fixed amount of computational effort. However, their approach requires information about the problem that is rarely available, such as noise intensity. Miller and Goldberg also investigated the effect of noise on different selection schemes, highlighting that the effect is similar to low selection pressure.

Rana et al. (1996) also argue that EAs ought to be robust to noise, due in part to being population-based. They found that noise had different effects on EAs depending on the level of noise and the actual fitness function itself. In fact, they showed that noise can actually be useful in the initial phases of the search. They also noted that increasing the sample size did not necessarily improve results.

[8]Resampling refers to sampling a (noisy) fitness function a number of times.

[9]Population sizing equations provide methods to estimate population sizes required for effective EA performance. The equations, however, require knowledge of problems that in practice is rarely available.

The works mentioned so far assume a fixed sample size for all individuals in the population and over all generations. Aizawa and Wah (1993, 1994) were perhaps the first to suggest adaptive sampling policies, where the number of samples is adapted over the course of an algorithm's execution. They recognised that if the difference between solutions is large, less accuracy is required to discriminate between them. They suggested two different strategies: increasing the sample size with the generation number or using a larger sample size for individuals with higher estimated variance.

Later Stagge (1998) investigated methods to reduce the required sample size for explicit averaging. For generational replacement, (μ, λ), and elitist, $(\mu + \lambda)$, selection, Stagge suggested that the sample size for evaluating individuals should be based on the probability of it being amongst the μ best, i.e. on its chance of surviving to the next generation. Hypothesis tests were used to evaluate the probabilities.

Rather than investigating methods to reduce the required resampling rate, several works, for example Sano and Kita (2000) and Branke et al. (2001), have suggested averaging fitness values over the neighbourhood of a point to be evaluated in order to improve fitness estimation. Rather than increasing the sample size to improve fitness estimations, which would require additional computational effort, models can be developed to improve the estimation using the evaluation of similar individuals that have already been discovered. However, both works make assumptions about the distribution of noise in neighbourhoods.

Boesel (1999) suggests that for linear ranking selection (and tournament selection), selection error can be reduced by grouping individuals of similar quality into one rank and assigning them the same selection probability. A corresponding mechanism for grouping based on statistical ranking and selection (R&S) procedures[10] is proposed. Furthermore, expanding on this work in (Boesel et al., 2003), a screening and selection "clean up" mechanism is suggested for use after the termination of search in order to select the best solution visited during the search with a prespecified confidence level.

To prevent selection error, Hedlund and Mollaghasemi (2001) combined an EA with a R&S procedure, specifically, an indifference-zone[11] selection procedure is used to select the new population by picking the best μ out of λ offspring, where $\lambda \geq \mu$, within an EA with a prespecified level of confidence. On termination, a "clean up"

[10]Statistical ranking and selection (R&S) procedures often used in Simulation Optimisation are methods for selecting the best system with some level of confidence out of a number of systems where their performance is uncertain. See, for example, Fu (2002); Kim and Nelson (2006) for more details.

[11]The indifference-zone is defined in terms of the maximum difference between solution evaluations that can be considered insignificant.

mechanism is also employed. Similarly, and based on Boesel et al. (2003), Buchholz and Thümmler (2005) proposed an indifference-zone R&S procedure for selection in a $(\mu+\lambda)$ EA, and maintain a pool of promising individuals throughout the run from which the best solution is selected on termination.

Branke and Schmidt (2003) proposed adapting stochastic tournament selection to take into account the inherent noise in the problem in order to determine the selection probability of the better individual based on the difference between the observed fitnesses of the tournament individuals. Secondly, they suggested using this idea to choose an appropriate resampling plan to discriminate between individuals for a specific tournament. Developing on their previous work, Branke and Schmidt (2004) proposed an elaborate sequential sampling strategy and compared it to a state-of-the-art indifference-zone R&S procedure, showing that both procedures can drastically reduce the number of samples required to guarantee a maximal selection error.

Also focussed on tournament selection, Cantú-Paz (2004) proposed a sequential sampling procedure using a hypothesis test to determine if the difference between the means of the two individuals chosen for a tournament is statistically significant. If not, then additional samples are allocated to the individual with the highest variance, and the test is carried out again. The procedure continues until the test is significant at which point the candidate with the highest fitness is the tournament winner. Cantú-Paz demonstrates that the sequential sampling procedure is superior to a fixed resampling rate or simply ignoring the noise (a single fitness function evaluation); however, the adaptive method can require a substantial amount of additional computation. In fact, for the problems tackled, using increased population sizes and direct comparison (no resampling) performed the best — although under, perhaps, unrealistic conditions: uniform noise across the space, and the availability of knowledge of problem-dependent parameters for population sizing models and noise levels.

Specifically focussed on search spaces where noise levels vary, Di Pietro et al. (2004) proposed several adaptive resampling methods to reduce noise in the estimation of fitness values. Firstly, a method to reduce the standard error below a pre-defined threshold is given, and, secondly, a more elaborate approach is provided involving specifying varying sample sizes for different levels of noise based on estimated standard deviations. They showed that when noise levels vary across the space it is better to dynamically resample for each point rather than use a fixed resampling rate for all points. However, they also concluded that applying the dynamic resampling approach is in general more complicated than they originally thought, and it can in fact perform worse than simply discriminating the noise into two different regions.

3.4.3 Schmidt's Framework: EAs and OCBA

The framework proposed by Schmidt et al. (2006) and (Schmidt, 2007) is, to the knowledge of the author, the first work to suggest a general framework for combining EAs with R&S procedures for problems in stochastic environments. In this symbiosis, the EA guides the search and the R&S method ameliorates the effects of noise on the efficacy of the heuristic. The proposed framework is applied to the problem addressed here and, due to this, an overview of the approach is given here and the technical details are presented in Section 5.4.1.

Based on the results from an extensive comparative study of state-of-the-art R&S procedures (Branke et al., 2007), Schmidt et al. (2006) and Schmidt (2007) proposed the tight integration of EAs and the Optimal Computing Budget Allocation (OCBA) procedure as a framework for tackling problems in stochastic environments. The OCBA, originally defined by Chen (1996), is a sequential R&S approach based on Bayesian statistics. The study showed that overall the OCBA was amongst the best R&S procedures in terms of a number of criteria, including the expected number of samples required to, say, achieve a given target probability, and how easy it is to use.

When integrated with an EA, OCBA is applied in each generation to provide a degree of confidence in the ordering of individuals based on their fitness values. More precisely, to guarantee with a predefined probability that the ranking information required for a particular EA (specifically, replacement and selection operators) to function effectively is correct. OCBA functions as an adaptive resampling scheme managing the efficient allocation of samples to individuals in order to determine the correct ranking of individuals. The power of OCBA is that it is based on an idea from Ordinal Optimisation[12] that in stochastic environments it is easier to rank candidate solutions than it is to precisely estimate their absolute performance.

As part of the framework, Schmidt et al. (2006) and Schmidt (2007) discuss the ranking information required by common EA variants. In particular, they show how common replacement operators, e.g. (μ,λ), $(\mu+\lambda)$ and generational replacement with elitism, and common selection operators, such as random selection, tournament selection and linear ranking, are supported in the framework. The R&S procedure itself, modified from vanilla OCBA, is adapted for the ranking of a set of individuals given

[12]See, for example, (Ho et al., 1992) for an introduction to Ordinal Optimisation.

the information required by the replacement and selection operators for the correct functioning of the EA. It is also noted that under these conditions only ranking-based selection schemes for EAs apply, because selection schemes that are based directly on fitness values, such as fitness proportionate methods, rely on accurate evaluations of fitness values.

For evaluation, the framework was compared on a single selection problem against various ranking configurations, including the complete ranking of all candidates, selecting the best from the set of candidate solutions, and an equal sampling allocation scheme that evaluates all individuals with an equal sample size for each stage of the OCBA. Here the single selection problem is meant to represent a single EA iteration. The configurations are compared on efficiency, total number of samples required, and accuracy, that is, whether the rankings are correct given the actual fitness values. The integrated approach is shown to be highly efficient with respect to the number of samples required. As mentioned, the framework is not, however, appraised over the entire run of an EA, where it is expected to be even more beneficial due to the re-use of samples of surviving individuals.

Furthermore, Schmidt (2007) discusses two distinct uses of EAs in stochastic environments, making the distinction between the EA as a 'Generator' and as an 'Optimiser'. As a generator of solutions, the EA is adapted for use in stochastic environments to account for selection error, so that the (estimated) best solution discovered at any point during the run is returned. In this mode the algorithm is used to generate and store points in the space and then, once the algorithm has terminated, to utilise a selection procedure to guarantee to some statistical confidence level that best solution of all the points visited is returned. Boesel et al. (2003) implemented this approach and used a screening procedure on termination of the EA to filter out obviously poor quality solutions, before selecting the best from the remaining set of candidates. Instead of storing all solutions, Buchholz and Thümmler (2005) proposed a heuristic method to maintain only an 'elite' population during the algorithm's run, and then, on termination, to apply a selection procedure to return the best from among the elite.

On the other hand, as an optimiser, the EA is used identically to the use in deterministic environments. Here the returned solution is the (estimated) best solution of the EA's population after the algorithm has terminated. In this thesis, EAs are viewed as optimisers; however, a method will be suggested to choose the best solution from the last population with a high level of confidence.

3.4.4 Conclusions

Despite a plethora of works proposing methodologies for handling optimisation problems in noisy and stochastic environments, there appears to be no clear guidelines on how to tackle such problems in general. Further to this, and perhaps more seriously, is that there is conflicting empirical evidence about which methodology to apply, in particular whether it is more beneficial to increase the resampling rate or the population size (see (Jin and Branke, 2005) for a summary of the debate). Furthermore, as pointed out by Bianchi et al. (2006b), empirical evidence is largely derived from ad-hoc continuous or discrete test functions, e.g. the sphere problem or the ONE-MAX problem. Hence empirical validation of the proposed approaches for applying EAs to SCOPs is still lacking. Nevertheless, as also pointed out by Bianchi et al. (2006b), and as highlighted above, a recent trend building on work in Simulation Optimisation[13] combining R&S procedures with EAs is showing promise. Schmidt's framework is perhaps testimony to this success; however, empirical evaluation of its overall effectiveness is still required.

3.5 EAs in Stochastic Combinatorial Optimisation

In this section, issues relating to the application of EAs to SCOPs are discussed, and related works from the literature are surveyed.

3.5.1 Introduction

Stochastic COPs (SCOPs) are combinatorial optimisation problems where all or part of the information about the problem data, for example, environment variables, is unknown, but where some knowledge about its probability distribution is assumed. Often SCOPs have deterministic counterparts, for example, the Travelling Salesman Problem (TSP), a COP, has as its SCOP counterpart the probabilistic TSP[14]. In general, due to the complexity of solving optimisation problems under uncertainty, which commonly involve calculations in high-dimensional spaces, classical approaches based on mathe-

[13] See (Fu, 2002) for a good overview of recent trends in Simulation Optimisation.

[14] The TSP is a canonical COP: given a set of customers (nodes) and distances between them (specified on edges between nodes), the goal is to find the minimum distance Hamiltonian cycle, that is, the cycle visiting all nodes exactly once with the minimum total distance. In the deterministic TSP it is required that all customers be visited at some point; in the probabilistic TSP (introduced by Jaillet (1985)) uncertainty is introduced so that customers have some probability of requiring a visit which we know only probabilistically when planning the trip.

matical and dynamic programming can only handle small problem instances and often require considerable computational effort. This provides an ideal scenario for the use of heuristic approaches that can find good quality, feasible, if not near-optimal, solutions in generally less computation time.

Bianchi et al. (2006b) provide a detailed survey of the use of metaheuristics, including EAs, in stochastic combinatorial optimisation. They include a discussion of various formulations of stochastic programming problems, of different approaches to evaluating objective functions, and a general discussion of some of the issues of the area.

In particular, they categorise three approaches for evaluating SCOP objective functions. The first is closed-form expressions, whereby analytical solutions are evaluated to provide exact objective values. Generally, if available, the closed-form expression will be evaluated directly to provide the exact objective value; however, for SCOPs, while it may be available, for example as a stochastic programming problem, it is often impractical to evaluate exactly.

The second approach is the use of ad-hoc and fast approximations of the problem objective, used when the exact evaluation is considered too time consuming during optimisation. The design of ad-hoc methods is strongly problem dependent, and no general rule exists for finding them. Examples include the use of a counterpart deterministic COP's objective function, truncating expressions within the objective function, or even using a set of scenarios instead of the true probabilistic model. Usually ad-hoc methods introduce a systematic error into the solution evaluation that can only be reduced through the design and use of a more accurate approximation.

The third and final approach involves estimating the objective value by simulation, for example using Monte Carlo sampling, where the value is the sample mean over a set of random samples. This approach introduces statistical error into the estimates of the candidate solutions making it impossible to discriminate (with finite resources) between solutions with certainty. The statistical error thus introduces a statistical selection problem whereby metaheuristics are often integrated with statistical ranking and selection mechanisms (as seen above). Due to the complexity and problem-dependent nature of first two classes, simulation approximation is perhaps the simplest method.

Bianchi et al. (2006b) also highlight several issues with drawing general conclusions from the literature of applying metaheuristics, including EAs, to optimisation under uncertainty, with the overall issue that there exists a schism between theory and practice. Firstly, to date, in general, the focus of evaluating approaches has been limited to continuous optimisation or using ad-hoc, artificial problems, e.g. the ONE-MAX problem. Real-world problems are often more complex and the search spaces less well understood. Secondly, of the several methods previously mentioned for evaluating objective functions, no general methodology has emerged. In fact, ad-hoc approaches are commonly used in applied works, despite being hard to design and to improve the accuracy of; in contrast with, for example, simulation approximation that can be improved by increasing sample sizes.

In the discussion section of the work, Bianchi et al. (2006b) considers issues with the use of simulation approximation. One of the issues concerns the trend of combining ranking and selection (R&S) methods with metaheuristics. In particular, they observe that such methods have been shown to have a great impact on the effectiveness of heuristic algorithms; nevertheless, it is unclear which of these methods is best when integrated with metaheuristics. Furthermore, another issue is whether the selection methods should be configurable, allowing for example an increase in sample sizes as the number of generations increases. Here they also mention that the application of variance reduction techniques could be fruitful in speeding up the convergence of sample estimates. A further point concerns the characterisation of noise. The relative size of the noise compared to the 'undisturbed' objective function values, or the 'degree of randomness', is an important factor. For example, it should not be expected that a metaheuristic will work well both small search spaces with high noise levels, where the focus might be on accurate objective function evaluation, and on large spaces with low noise, where the focus might be on exploring the search space more than accurately estimating fitness values.

3.5.2 Applications

We now provide a survey of existing works addressing the application of EAs to SCOPs. Due to the wide range of problems tackled, the survey focusses on four general points: the specific problem tackled; the algorithm employed; how the objective function is evaluated; and, finally, how the approach is evaluated.

A number of works have applied EAs to SCOPs, including (Easton and Mansour, 1999; Watson et al., 1999; Jellouli and Chatalet, 2001; Yoshitomi et al., 2000; Yoshitomi, 2002; Yokoyama and Lewis III, 2003; Mak and Guo, 2004; Bianchi et al., 2004, 2006a; Sudhir Ryan Daniel and Rajendran, 2005; Opasanon, 2004; Miller-Hooks and Sorrel, 2008; Opasanon and Miller-Hooks, 2010). However, only (Opasanon, 2004;

Miller-Hooks and Sorrel, 2008; Opasanon and Miller-Hooks, 2010) address SCOPs that are relevant to the problem addressed herein and hence these are discussed in detail. For reviews of the other works, see Bianchi et al. (2006b).

3.5.2.1 Minimum Time Network Flow Problem

Opasanon (2004) developed a metaheuristic framework based on Genetic Algorithms for solving a priori flow problems in STV Networks. The framework was first applied to two problems in time-dependent networks with stochastic, time-varying capacities but deterministic, time-varying travel times, namely, a variant of the Time-Dependent Quickest Flow problem (TDQFP) (variation of (Miller-Hooks and Patterson, 2004)) and the Safest Escape Problem (see 2.5.4). In these problems waiting is not permitted. Next, the framework was extended to tackle flow problems in STV Networks, that is, in network models in which travel times are stochastic too; specifically, the minimum time dynamic flow problem and multi-objective problems (Opasanon, 2004; Opasanon and Miller-Hooks, 2010), where waiting is allowed. We now outline the framework and discuss its evaluation.

In Opasanon's framework, each chromosome represents a flow pattern, where each gene is a pair consisting of a path (represented by a set of edges each with an associated departure time) and an associated flow. Different chromosomes can therefore have different numbers of genes, depending on the number of paths and associated flow levels they contain. Problem constraints are handled through the solution representation and genetic operators, so that only feasible solutions are generated. Initialisation ensures that flow conservation is guaranteed by generating new solutions through iteratively finding random paths from the source to sink and assigning a random amount of flow from the residual amount to the path, until all supply is allocated. For crossover, two parents are selected and for each departure time their genes are either ranked by their ratio of path cost, e.g. travel time, to path capacity, or alternatively, given some crossover rate, simply randomly selected. If ranked, the paths with the highest ratio are chosen and a random amount of flow is assigned according to the supply available and the capacity of the path (determined by the minimum capacity found among all the edges in the path). This procedure is carried out until all flow is allocated. Mutation involves replacing with some probability the last gene for each departure time in a chromosome by a set of new paths and flow that accommodates enough flow. For selection, a binary tournament operator is utilised with elitism, where the fitness value of each solution is based on the objective function for the specific problem. Since optimal solutions were available for the two problems considered, termination criteria were met when either a number of generations had passed or convergence to an optimal solution was achieved. The framework was empirically evaluated for each of the two problems against the results from exact algorithms. For the variant of the TDQFP, the GA was shown to find solutions on average within 5% of the optimal, while for the SEscape problem the GA approach found optimal solutions or near-optimal solutions.

The framework was then extended to solve the Minimum Time Dynamic Flows (MTDF) problem in STV Networks. In STV Networks, edge travel times are now also stochastic, so that it is no longer possible to know a priori with certainty where flow will be in the network at any given time (as it was previously). To cope with this increased complexity, the framework was further developed using the concept of Noisy Genetic Algorithms (NGAs).

Due to the complexity of evaluating candidate solutions in STV Networks, a noisy fitness function is defined to estimate the expected performance of candidates on the network. Evaluating the exact performance of candidates would require evaluation across all possible network states, where a network state is a single realisation of each of the random values (corresponding to edge properties) in the network. However, in general this is not practical since the number of network states grows exponentially in the number of edges, the time horizon, and the size of the support for edge capacity and travel time distributions. To handle this, a simulation approximation approach was developed, where in each generation individuals are evaluated on a single random sample of network states in order to estimate their expected performance. Since the same sample is used to evaluate all individuals of a generation, the evaluations are correlated[15]. Furthermore, the variance-reduction technique *stratified sampling*[16] is employed in order to include rarer but large impact realisations of edge properties in the sampled states. Additionally, an adaptive sampling plan is used based on that of Smalley (1998) and Gopalakrishnan et al. (2001).

Due to the increased complexity of the problems in STV Networks, where it is unlikely that in general solutions exist that are feasible on all network states, infeasible solutions are now allowed in the population, and hence several changes were made to the framework to handle this. Initialisation is now carried out on the most likely state (MLS) of the network, in order to guarantee that the first generation is at least

[15]Correlated sampling is a well-known variance reduction technique, see, for example, Law and Kelton (2000) for more details.

[16]For details on the stratified sampling technique, see, for example, Bratley et al. (1987).

feasible on the most probable realisation of the network (however unlikely it may be). The same method as before is used for generating the initial population. In evaluation, a penalty function is employed to penalise candidate solutions that violate problem constraints. For the MTDF problem, flow waiting at a node due to a lack of available onward capacity incurs additional cost, or, in other words, forced waiting due to capacity restrictions acts as a penalty. Finally, given these changes, the heuristic algorithm is evaluated against an exact procedure for the deterministic MTDF on a network with 27 states. The exact algorithm's performance is given as the weighted average of the result for each of the network states. The NGA is reported to find a solution with a difference of 4.26% from the result of the exact procedure. Note, however, that the evaluation of the algorithm on a single problem instance does not provide a thorough empirical evaluation of the approach, and hence it is difficult to gauge its overall effectiveness.

3.5.2.2 Maximal Dynamic Expected Flows Problem

Miller-Hooks and Sorrel (2008) applied the NGA framework from (Opasanon, 2004) to the Maximal Dynamic Expected Flows (MDEF) problem in STV Networks. The MDEF problem seeks the flow pattern that maximises the expected number of units that successfully reach the sink within a time bound. Once again, due to the complexity of the flow problem, a heuristic approach using simulation approximation to evaluate individuals was employed.

The framework of Opasanon was adapted for the MDEF problem, and we now discuss the modifications. The basic representation used by Opasanon was extended: a candidate solution is now defined to be a vector (ordered list) of chromosomes (as defined by Opasanon (2004)), one for each departure time from the source. Similarly to the previous work, the initial population was randomly generated on the basis of the most likely state (MLS), but now the MLS is also used in crossover and mutation. If it is not possible to generate the required number of initial (feasible) solutions on the MLS then the problem is deemed too difficult and the algorithm aborts.

For mating selection, parents are chosen using fitness-proportionate selection. Then, for crossover, the genes of two parents are once again ordered by capacity to travel time ratios when evaluated on the MLS. However, instead of choosing strictly based on the ratios, as previously done, a 'roulette wheel' is used to pick paths with a probability proportional to their ratios. Flow is then assigned based on available supply on the MLS. Mutation is now applied to the extended chromosome, so that each chromosome contained in the extended chromosome for each departure time has a chance of getting mutated. Then, as before, with a predefined probability the last gene of the chromosome is replaced by a set of new paths covering the same flow. Evaluation is also implemented using a noisy fitness function that evaluates each candidate solution on a sample of the same 20 network states in order to estimate their expected performance[17]. Again, the evaluation of solutions is correlated due to this use of the same sample of states. An elitist $(\mu + \lambda)$ selection is used as a replacement strategy. Finally, the NGA approach is evaluated on a set of randomly generated instances and the performance of the algorithm is analysed under different parameter configurations. No baseline comparison is carried out or bounds provided so that it is difficult to evaluate the overall effectiveness of the approach.

3.5.3 Conclusions

This section has discussed the application of EAs to SCOPs. It began with a discussion of the general issues, outlining the various approaches to evaluating objective functions, discussing the gap between theory and practice, and that there is at present no clear methodology for approaching SCOPs. This section also surveyed the application of EAs to a number of SCOPs, including several flow problems in STV Networks closely related to the problem addressed herein.

3.6 Summary

The literature review chapters have covered a wide array of works, and whereas the previous chapter focussed on related optimisation problems for evacuation planning, this chapter has explored the solution approach, Evolutionary Algorithms, proposed for solving the problem addressed herein.

The chapter began with an overview of EAs and their common variants, and then presented a generic EA and discussed the various components of the algorithm. It highlighted in particular the use of EAs for combinatorial and constrained optimisation.

Next it discussed the application of EAs to a number of related combinatorial optimisation problems, including shortest path and network flow problems. The discussion

[17]Miller-Hooks and Sorrel (2008) note a practical issue with evaluating extremely small probabilities represented by floating-point numbers. The issue arises due to their explicit need to evaluate the probability of individual network states and to normalise over sets of sampled network states. This issue does not occur in the work described herein because probabilities are not explicitly required by the chosen Monte-Carlo sampling method.

concluded that EAs have not been extensively applied to such problems, perhaps due to the availability of exact and efficient algorithms; nevertheless, as problem sizes grow and additional constraints are required, the use of EAs becomes more appropriate.

A substantial review of EAs in noisy environments is then provided, with a focus on noisy fitness evaluations, that is, where the evaluation of individuals is uncertain and unascertainable with finite resources. The impact of noise on the algorithms is expounded and a number of works addressing these issues are reviewed. In particular, a recent framework, which is applied to the optimisation problem addressed herein, proposing the tight integration of EAs with a state-of-the-art statistical ranking and selection method is introduced. The framework is one of a number of recent researches suggesting the combined use of EAs and statistical ranking and selection methods for addressing noisy fitness evaluations. The review also highlighted the conflicting evidence that suggests different approaches for tackling problems in noisy environments.

Finally, a review of the use of EAs in stochastic combinatorial optimisation is provided. The section began by discussing a number of important issues arising due to the application of EAs to SCOPs, including different methods for approximating objective functions, the gap between theory and practice, and the lack of clear methodology for approaching SCOPs in general. Furthermore, a number of applications of EAs to SCOPs are surveyed, with focus on several works proposing EAs to solve flow problems in STV Networks that are closely related to the problem addressed herein.

Chapter 4

Maximal Safest Escape Problem

This chapter formally defines and explicates the core research problem. Firstly, it provides the necessary formal notation and preliminaries for STV Networks. Secondly, it discusses various route selection criteria for problems in STV Networks. Then, thirdly, it states the mathematical formulation of the Maximal Safest Escape Problem in STV Networks, including a discussion on the combinatorics and computational complexity of the problem. Finally, it lays out a procedure for providing stochastic upper bounds for optimal solutions to the Maximal Safest Escape problem.

4.1 Notation and Preliminaries

We now provide the notation and preliminaries used to formally define time-dependent flow networks with edge capacities and travel times that are both stochastic and time-varying. This model is based on the definitions given by Opasanon (2004), Opasanon and Miller-Hooks (2008) and Miller-Hooks and Sorrel (2008).

4.1.1 Basic Definitions

Let a time-dependent network $\mathscr{N} = (\mathscr{G}, \mathscr{M}, \mathscr{B}, \mathscr{T}, \mathscr{R})$ consist of a finite directed graph, $\mathscr{G} = (\mathscr{V}, \mathscr{E}, [T])$, where $\mathscr{V}$ is the set of nodes, $v = |\mathscr{V}|$, $\mathscr{E} \subseteq \mathscr{V}^2$ is the set of directed edges connecting nodes, $e = |\mathscr{E}|$, and $[T] \equiv \{0, 1, \ldots, T\}$ is the time frame of interest.

In these networks, edge properties are modelled as discrete-time stochastic processes. Specifically, edge capacities are represented as discrete random variables with probability mass functions given by the set $(\mathscr{M}, \mathscr{B})$. Associated with each edge

$(i,j) \in \mathscr{E}$ at time $t \in [T]$ is a set of $D \in \mathbb{N}$ non-negative, integer-valued, time-varying capacities, $\mathscr{M} = \{\mu_{ij}^{d}(t)|d = 1,\dots,D\}_{(i,j)\in\mathscr{E},t\in[T]}$, with corresponding probabilities $\mathscr{B} = \{\beta_{ij}^{d}(t)|d = 1,\dots,D\}_{(i,j)\in\mathscr{E},t\in[T]}$. Similarly, travel times are described by discrete random variables with probability mass functions that vary with time, given by the set $(\mathscr{T},\mathscr{R})$. For each edge $(i,j) \in \mathscr{E}$ at time $t \in [T]$ is a set of $H \in \mathbb{N}$ non-negative, integer-valued, time-varying travel times, $\mathscr{T} = \{\tau_{ij}^{d}(t)|h = 1,\dots,H\}_{(i,j)\in\mathscr{E},t\in[T]}$ and corresponding probabilities $\mathscr{R} = \{\rho_{ij}^{h}(t)|h = 1,\dots,H\}_{(i,j)\in\mathscr{E},t\in[T]}$. For simplicity, travel times are assumed to be multiples of discretised time intervals. The uncertain edge properties are assumed to be realised upon entrance to an edge at a departure time, and fixed for those units of flow departing from the source node of the edge (so-called frozen-link property (Orda and Rom, 1990)). The flow on edge $(i,j) \in \mathscr{E}$ that leaves node i at departure time $t \in [T]$ is given by the function $x_{ij}(t) : \mathscr{E} \times [T] \to \mathbb{Z}$. The set of predecessor and successor nodes of a node $i \in \mathscr{V}$ are given by $\Gamma^{-1}(i) = \{j|(j,i) \in \mathscr{E}\}$ and $\Gamma^{+1}(i) = \{j|(i,j) \in \mathscr{E}\}$, respectively.

Note that only during the time period of interest, $t \in [T]$, may edge attributes vary with time, so that $\forall t > T$ it is assumed that the properties are static, taking the same values as at the last time interval T. Finally, a simplifying assumption is that travel times and capacities are assumed to be independent over time and space.

4.1.2 Single Source, Single Sink

In the networks, we identify two special nodes, a source node, $s \in \mathscr{V}$, and a sink node, $l \in \mathscr{V}$, and we assume $s \neq l$ and $(s,l) \notin \mathscr{E}$, and also $\Gamma^{-1}(s) = \emptyset$ and $\Gamma^{+1}(l) = \emptyset$. A general and efficient algorithm by Miller-Hooks and Patterson (2004) can convert networks with multiple sources and sinks to equivalent single source-sink networks.

4.1.3 First-In-First-Out

Modelling of edges requires consideration of the ordering by which units can travel along edges; that is, for an edge (i,j) units must arrive at the target node of an edge, j, in the order that they left the edge head node, i. Problems that enforce First-In-First-Out (FIFO) conditions are generally regarded as easier to solve, but perhaps less realistic. A more general modelling decision is to allow edges to be non-FIFO, so that for an edge the order in which units arrive at the target node is not necessarily the order in which they left the head node. In this work, non-FIFO is allowed.

4.1.4 Supply/Demand Model

The supply available at a node is the amount of flow that is ready to be shipped along some outgoing edge at a time t. For a negative amount, this is considered a demand, or a required level of supply. The supply/demand at a node i at departure time t is given by the function $b_i(t) : \mathscr{V} \times [T] \to \mathbb{Z}$. At the source node, s, the supply is $b_s(t) \in \mathbb{N}_0, \forall t \in [T]$; that is, supply at the source is defined in terms of non-negative integer values for any departure time. For convenience, let the set of non-zero supply times be $S = \{t \in [T] | b_s(t) > 0\}$. At the sink node, l, $b_l(t) = 0, \forall t \in [T-1]$, and, $b_l(t = T) \in \mathbb{Z}^-$. Also, it is assumed there is no supply beyond the time horizon, $b_i(t) = 0, \forall i \in \mathscr{V}, \forall t > T$. At time T then, $b_l(T) = -\sum_{t \in S} b_s(t) = -B$, that is, the demand of the network will be equal to the negated total supply B. Thus, if flow arrives prior to time T, it simply waits without penalty (implicit waiting) until time T in order to fulfil the demand. Units are not allowed to wait at the source or intermediate nodes, that is, $\forall i \in \mathscr{V} \backslash \{l\} : (i,i) \notin \mathscr{E}$. Finally, $b_i(t) = 0, \forall i \in \mathscr{V} \backslash \{s,l\}, \forall t \in [T]$.

In summary,

$$b_s(t) \in \mathbb{N}_0, \forall t \in [T]$$

$$b_i(t) = \begin{cases} 0 & \forall i \in \mathscr{V} \backslash \{s,l\}, \forall t \in [T] \\ 0 & \forall i \in \mathscr{V}, \forall t > T \end{cases}$$

$$b_l(t) = \begin{cases} 0 & \forall t \in [T-1] \\ -B & t = T \end{cases}$$

4.1.5 Paths and Flows

A directed $s-l$ path in a network $\mathscr{N}$ is a sub-graph of the topological graph $\mathscr{G}$, denoted by σ, consisting of a sequence of n nodes: $\sigma = \left(i_1 = s, i_2, \ldots, i_{n-1}, i_n = l\right)$, with the property that $1 \le k < n$, $(i_k, i_{k+1}) \in \mathscr{E}$. The directed $s-l$ paths are assumed to be simple, so that they contain no repetition of nodes (cycles). Henceforth, directed simple $s-l$ paths are called paths. The length of a path, σ, with n nodes is equal to the number of edges in the path, $n-1$. The set of all paths in a network $\mathscr{N}$ is denoted Ω.

A flow pattern in a network $\mathscr{N}$ is a multiset[1] $\psi = (R, m)$, where $R = \Omega \times [T]$ is the set of underlying elements, consisting of the set of pairs: $\{(\sigma, t) | \sigma \in \Omega, t \in [T]\}$, and $m : \Omega \times [T] \to \mathbb{N}_0$ is the multiset multiplicity function. Each pair (σ, t) represents a path $\sigma \in \Omega$ and departure time $t \in [T]$ from the source s to the sink l, and the function

[1]A multiset, or bag, is a generalisation of a set where elements can have a membership of more than one.

$m(r)$, $r \in \Omega \times [T]$, specifies the total number of units of flow assigned to the pair r. If the multiplicity of $r \in R$ is 0, $m(r) = 0$, then 0 units of flow are assigned to the pair r. The cardinality of a flow pattern is given by $|(R,m)| = \sum_{r \in R} m(r)$. The set of all flow patterns for a network $\mathscr{N}$ is denoted Ψ.

4.1.6 Network States

A network state represents a single realisation of all random elements — all edge travel times and capacities over time. In the a priori context, the network state to be realised is of course as yet unknown; however, the set of possible network states can be determined from the joint distribution of all edge travel time and capacity distributions for all network edges over time. A network $\mathscr{N}$ can realise to a finite set of mutually exclusive states, ξ^i, $i = 1, \ldots, k$, denoted by Ξ, where $k = (D \cdot H)^{e(T+1)}$. The random event that models the uncertain network state is denoted by $\bar{\xi}$.

We have a probability distribution over Ξ, $\mathbb{P}$, where, given the independence of edge properties,

$$\begin{aligned}\mathbb{P}(\bar{\xi} = \xi) &= P\Big(\bigcap_{\substack{(i,j) \in \mathscr{E} \\ t \in [T]}} \bar{\mu}_{ij}(t) = \mu^{\xi}_{ij}(t) \cap \bar{\tau}_{ij}(t) = \tau^{\xi}_{ij}(t) \Big) \\ &= \prod_{\substack{(i,j) \in \mathscr{E} \\ t \in [T]}} P\big(\bar{\mu}_{ij}(t) = \mu^{\xi}_{ij}(t)\big) \cdot P\big(\bar{\tau}_{ij}(t) = \tau^{\xi}_{ij}(t)\big)\end{aligned}$$

where $\bar{\mu}_{ij}(t)$ and $\bar{\tau}_{ij}(t)$ are random variables for the yet unrealised capacity and travel time values, respectively, for edge (i,j) at time t, and $\mu^{\xi}_{ij}(t)$ and $\tau^{\xi}_{ij}(t)$ the (hypothetically) realised capacity value and travel time value, respectively, for edge (i,j) at time t for state $\xi \in \Xi$. For convenience, we abbreviate $\mathbb{P}(\bar{\xi} = \xi)$ to $\mathbb{P}(\xi)$. Then, by mutual exclusivity, for some $X \subseteq \Xi$,

$$\mathbb{P}(X) = \bigcup_{\xi \in X} \mathbb{P}(\xi) = \sum_{\xi \in X} \mathbb{P}(\xi)$$

Given a flow pattern, in general a network state is feasible if it does not violate any problem constraints. Let the feasible set of states given a flow pattern $\psi \in \Psi$ be denoted $\mathcal{F}_\psi \subseteq \Xi$. Conversely, $\bar{\mathcal{F}}_\psi$ denotes the set of infeasible states, so that $\Xi = \mathcal{F}_\psi \cup \bar{\mathcal{F}}_\psi$; of course states are either feasible or infeasible for a given flow pattern $\psi \in \Psi$, $\mathcal{F}_\psi \cap \bar{\mathcal{F}}_\psi = \emptyset$.

In order to discuss the feasibility of candidate solutions a priori, that is, while the network state is uncertain, flow patterns are referred to as:

infeasible If $\mathcal{F}_\psi = \emptyset$ then $\mathbb{P}(\mathcal{F}_\psi) = 0$.

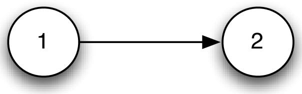
p-feasible If $\mathcal{F}_\psi \subseteq \Xi$ then $\mathbb{P}(\mathcal{F}_\psi) \in (0,1]$.

feasible If $\mathcal{F}_\psi = \Xi$, then $\mathbb{P}(\mathcal{F}_\psi) = 1$ almost surely. (Candidates that are feasible are also p-feasible.)

4.1.7 Example

In this section an example is provided to illustrate the notation and preliminaries introduced in the previous sections.

Figure 4.1 illustrates an example STV Network edge between two nodes: 1 and 2, with edge properties for departure time $t = 1$. For the edge $(1,2)$, for example, the probability that it can support 3 units of supply at departure time $t = 1$ is $P(\bar{\mu}_{12}(1) \geq 3) = 0.6$. The probability of arriving at node 2 at time $t' = 4$, departing node 1 at time $t = 1$, is equal to $P(\bar{\tau}_{12}(1) = 4 - 1 = 3) = 0.8$.



$\mu^2_{12}(1) = 3$	$\beta^2_{12}(1) = 0.6$
$\mu^1_{12}(1) = 1$	$\beta^1_{12}(1) = 0.4$
$\tau^2_{12}(1) = 5$	$\rho^2_{12}(1) = 0.2$
$\tau^1_{12}(1) = 3$	$\rho^1_{12}(1) = 0.8$

Figure 4.1: Example STV Network Edge

Figure 4.2 and Tables 4.1 and 4.2 describe an example STV Network. Properties of the network can be seen in Tables 4.1, 4.3 and 4.4. In the network, there are 3 paths and, given the supply structure, 9 flow patterns; however, the quality and feasibility of these flow patterns depends on the specific problem to be solved. This example is revisited after the problem definition (Section 4.3) to provide an example problem instance with solutions.

The previous sections have detailed the necessary notation and preliminaries for describing STV Networks, and illustrated their use with an example. The following sections build upon this and look to define a suitable optimisation problem for the domain of interest under the conditions of transiency and uncertainty.

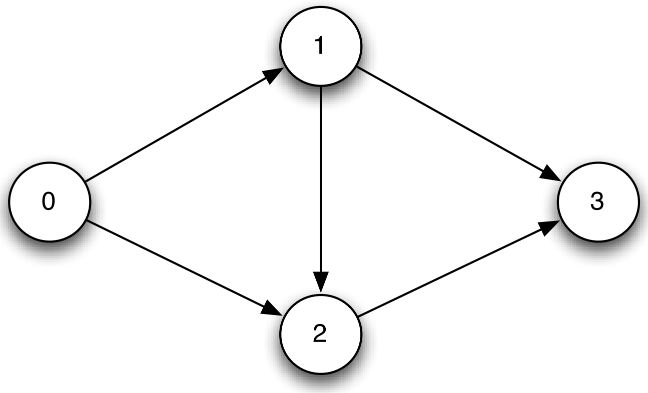


Figure 4.2: Example STV Network: Topological Graph

Property	Values
Nodes	$\mathscr{V} = \{0,1,2,3\}$
Edges	$\mathscr{E} = \{(0,1),(0,2),(1,2),(1,3),(2,3)\}$
Time Horizon	$T = 6$; $[T] = \{0,1,\ldots,6\}$
Source Node	$s = 0$
Sink Node	$l = 3$
Network Supply	$b_s(0) = 1, b_s(1) = 1$; $B = 2$
Paths	$\Omega = \{\sigma_1 = (0,1,3), \sigma_2 = (0,1,2,3), \sigma_3 = (0,2,3)\}$
Flow Patterns	$\Psi = \{\psi_1,\psi_2,\psi_3,\psi_4,\psi_5,\psi_6,\psi_7,\psi_8,\psi_9\}$ (See Table 4.2)
Network States	$\lvert\Xi\rvert = (2\cdot 2)^{5\cdot(6+1)} = 4^{35} \approx 1.18 \cdot 10^{21}$

Table 4.1: Example STV Network: Network Properties

4.2 Route Selection Criteria

Based on the model of STV Networks defined above, the following sections discuss possible route selection criteria relevant to the domain.

4.2.1 Status Quo

As discussed in the literature review, in Section 2.3, *time*, in measures such as the total evacuation time or the last time to exit (SFPE, 2002), is currently used as the principle route selection criterion in the development of emergency movement plans. In these contexts, time acts as a proxy for measuring the exposure to danger of persons involved

$\psi_1 = \big(R = \{(\sigma_1, t_0), (\sigma_1, t_1)\}, (m((\sigma_1, t_0)) = 1, m((\sigma_1, t_1)) = 1)\big)$
$\psi_2 = \big(R = \{(\sigma_1, t_0), (\sigma_2, t_1)\}, (m((\sigma_1, t_0)) = 1, m((\sigma_2, t_1)) = 1)\big)$
$\psi_3 = \big(R = \{(\sigma_1, t_0), (\sigma_3, t_1)\}, (m((\sigma_1, t_0)) = 1, m((\sigma_3, t_1)) = 1)\big)$
$\psi_4 = \big(R = \{(\sigma_2, t_0), (\sigma_1, t_1)\}, (m((\sigma_2, t_0)) = 1, m((\sigma_1, t_1)) = 1)\big)$
$\psi_5 = \big(R = \{(\sigma_2, t_0), (\sigma_2, t_1)\}, (m((\sigma_2, t_0)) = 1, m((\sigma_2, t_1)) = 1)\big)$
$\psi_6 = \big(R = \{(\sigma_2, t_0), (\sigma_3, t_1)\}, (m((\sigma_2, t_0)) = 1, m((\sigma_3, t_1)) = 1)\big)$
$\psi_7 = \big(R = \{(\sigma_3, t_0), (\sigma_1, t_1)\}, (m((\sigma_3, t_0)) = 1, m((\sigma_1, t_1)) = 1)\big)$
$\psi_8 = \big(R = \{(\sigma_3, t_0), (\sigma_2, t_1)\}, (m((\sigma_3, t_0)) = 1, m((\sigma_2, t_1)) = 1)\big)$
$\psi_9 = \big(R = \{(\sigma_3, t_0), (\sigma_3, t_1)\}, (m((\sigma_3, t_0)) = 1, m((\sigma_3, t_1)) = 1)\big)$

Table 4.2: Example STV Network: Flow Patterns (pairs with multiplicity of 0 are excluded for readability)

	$(\mu^d_{ij}(t), \beta^d_{ij}(t))$	
(i,j)	$t \leq 3$	$t \geq 4$
$(0,1)$	$\binom{2}{0}\ \binom{0.6}{0.4}$	$\binom{1}{0}\ \binom{0.2}{0.8}$
$(0,2)$	$\binom{3}{1}\ \binom{0.6}{0.4}$	$\binom{3}{1}\ \binom{0.6}{0.4}$
$(1,2)$	$\binom{5}{3}\ \binom{0.3}{0.7}$	$\binom{5}{3}\ \binom{0.2}{0.8}$
$(1,3)$	$\binom{3}{1}\ \binom{0.6}{0.4}$	$\binom{2}{0}\ \binom{0.4}{0.6}$
$(2,3)$	$\binom{2}{0}\ \binom{0.6}{0.4}$	$\binom{1}{0}\ \binom{0.8}{0.2}$

Table 4.3: Example STV Network: Network Edge Capacities

	$(\tau^h_{ij}(t), \rho^h_{ij}(t))$	
(i,j)	$t \leq 2$	$t \geq 3$
$(0,1)$	$\binom{2}{1}\ \binom{0.5}{0.5}$	$\binom{2}{1}\ \binom{0.7}{0.3}$
$(0,2)$	$\binom{2}{1}\ \binom{.99}{.01}$	$\binom{2}{1}\ \binom{0.7}{0.3}$
$(1,2)$	$\binom{3}{2}\ \binom{0.2}{0.8}$	$\binom{3}{2}\ \binom{0.8}{0.2}$
$(1,3)$	$\binom{3}{1}\ \binom{0.6}{0.4}$	$\binom{3}{1}\ \binom{0.6}{0.4}$
$(2,3)$	$\binom{3}{1}\ \binom{0.4}{0.6}$	$\binom{3}{1}\ \binom{0.4}{0.6}$

Table 4.4: Example STV Network: Network Edge Travel Times

in emergency incidents, so that by minimising the exposure to potentially hazardous events the chance of harm is minimised. However, this need not always be the case: for example, the fastest path is not necessarily the safest. In the context of future emergency-response systems, given the necessary infrastructure, it may be possible to consider the conditions of the environment more directly, whereby a more informative model of the risks are available to decision makers.

4.2.2 Performance Criteria in STV Networks

The previous chapter surveyed a number of flow problems from the literature that are typically applied to evacuation scenarios; for example, the quickest time problem and universal maximum flow problem. These problems have been investigated mostly in time-dependent or time-varying flow networks[2]. In stochastic environments, it is necessary to work with the uncertainty in network elements, which in general makes the flow problems more complex. For example, in time-varying flow networks with stochastic edge properties, in particular edge capacities, it is no longer possible to know *a priori* capacity values with certainty. Hence for different realisations of the network, different numbers of people may be able to pass through a given passageway.

As discussed by Opasanon (2004), there are several ways in which to handle uncertainty. One common approach is to model the uncertainty using random variables and to work with *expectations*. A simple approach is to eliminate the uncertainty by transforming a stochastic problem to a deterministic one, via, for example, the Certainty Equivalent heuristic, which replaces random variables with their expected values. Loui (1983) showed that using this conversion for stochastic, time-invariant networks allows for the optimal solution to be found on the transformed network; whereas Hall (1986) demonstrated that for stochastic, time-dependent networks this conversion will not yield the expected value for a path because the travel time (and any other time-varying property) on an edge depends on the arrival time, which is uncertain, at the head node of each edge in the path. In any case, an issue with *expectations* is that on realisation the actual value may be significantly less or more than the expected value — it may have a large variance. The consideration of these events could be important in the emergency domain. Opasanon discussed several approaches, focussing on stochastic edge capacities, that address this issue, such as the expected flow of an edge, and

[2]As a reminder, time-dependent flow networks model flow as a function of time, whereas time-varying models additionally model network properties such as supply and edge properties also as functions of time.

the probability of successful traversal of an edge, on which the Safest Escape Problem is based as discussed previously in 2.5.4.

Others have handled the uncertainty through the consideration of network performance measures. The maximum expected flow, for example, is a performance measure for flow networks. It can be used to determine, for example, the likelihood that certain level of flow is sustainable on a network. However, in general this measure describes the asymptotic behaviour of the system, that is, it does not provide details of the behaviour for a particular realisation but over many runs, and thus may have limited applicability in the emergency domain. Other common, related measures include network reliability and connectedness that also investigate performance measures of networks. For example, network reliability finds the probability that flow networks can accommodate certain levels of supply. An issue with these measures, as noted previously, is that they do not develop routing plans; they act solely as a general measure of performance that, again, may not be particularly suitable for the emergency domain.

The optimisation problem formulated herein takes a hybrid approach, similar to the Safest Escape Problem (2.5.4), combining the evaluation of a performance measure with the development of routing plans. Specifically, in STV Networks, given the time-varying and stochastic nature of edge properties, the focus is on the feasibility of flow patterns given the possible realisations of the network elements. For example, the flow pattern with the maximal probability of feasibility for all potential realisations of the network is one measure of the performance of the system that could perhaps be useful in the emergency domain. In other words, in the context of emergency evacuation in the built environment, it is the set of routing instructions with the highest overall chance for building occupants to successfully egress through building circulation systems, given the current environmental conditions and perhaps future predicted scenarios, from the occupants' starting locations to places of safety.

A number of route selection criteria are available for handling the uncertainty in stochastic flow networks, from utilising expected values of the random variables that model the uncertainty for developing routing plans, to using network performance measures as system-wide measures of the state of the network. The work described herein defines and addresses a problem based on a dual approach, developing routing plans and also working as a measure of network performance.

4.3 Maximal Safest Escape Problem

In this section, the problem, including the problem objective and constraints are defined, which is followed by a redefinition of the objective as an *expectation*. Next, the problem is shown to be NP-hard, and, finally, a discussion on the problem combinatorics is provided.

4.3.1 Problem Definition

The Maximal Safest Escape (MSE) problem is defined as the problem of finding *a priori* flow patterns with the maximum probability of feasibly shipping the available supply from the network source to the network sink in STV Networks.

The objective function is defined as:

$$\max_{\psi \in \Psi} \mathbb{P}(\mathcal{F}_\psi) \tag{4.1}$$

Additionally, flows are subject to the following deterministic constraints (from (Miller-Hooks and Patterson, 2004)), that is, constraints that apply to realised network states:

$$\sum_{j \in \Gamma^{+1}(i)} x_{ij}(t) - \sum_{j \in \Gamma^{-1}(i)} \sum_{\{t' | t' + \tau_{ji}(t') = t\}} x_{ji}(t') = b_i(t), \forall i \in \mathscr{V}, \forall t \in [T], \tag{4.2}$$

$$0 \leq x_{ij}(t) \leq \mu_{ij}(t), \forall (i,j) \in \mathscr{E}, \forall t \in [T] \tag{4.3}$$

To ensure that the supply/demand structure of the network is maintained, flow conservation constraints (4.2) are included. Also, non-negativity and capacity constraints (4.3) are included to ensure that the flow on edges does not exceed the capacity and is not negative.

4.3.1.1 Definition as an Expectation

It is possible to define the objective function as an expectation using the indicator function. In this way, the objective is more conducive to approximation methods.

Define the indicator function for some $X \subseteq \Xi$ as $I_X : \Xi \rightarrow \{0,1\}$, where:

$$I_X(\xi) = \begin{cases} 1 & \xi \in X \\ 0 & \xi \notin X \end{cases}$$

For a set of network states, the indicator function returns 1 if the state is a member of the set of states, or 0, if it is not.

Then the MSE defined in terms of the expectation of the indicator function is:

$$\begin{aligned}\mathbb{P}(\mathcal{F}_\psi) &= \sum_{\xi \in \mathcal{F}_\psi} \mathbb{P}(\xi) \\ &= \sum_{\xi \in \Xi} I_{\mathcal{F}_\psi}(\xi) \cdot \mathbb{P}(\xi) \\ &= \mathbb{E}[I_{\mathcal{F}_\psi}(\bar{\xi})]\end{aligned}$$

Say $h(\psi, \bar{\xi}) = I_{\mathcal{F}_\psi}(\bar{\xi})$, then

$$\mathbb{E}[h(\psi, \bar{\xi})] = \mathbb{E}[I_{\mathcal{F}_\psi}(\bar{\xi})] \tag{4.4}$$

and we can write the definition in the more traditional stochastic optimisation form, where the objective function has the goal of maximising the expectation:

$$\max_{\psi \in \Psi} \left[G(\psi) = \mathbb{E}[h(\psi, \bar{\xi})] \right] \tag{4.5}$$

We denote the objective value of a solution ψ by $G(\psi)$, and the optimal solution $\psi^* = \arg\max_{\psi \in \Psi} G(\psi)$ with $z^* = G(\psi^*)$.

4.3.2 Example

Based on the example STV Network and supply structure described in Section 4.1.7, we now provide an illustrative example of the MSE problem.

Given the example described previously, the goal of the MSE is to find flow patterns with the highest likelihood of successfully conveying the supply from the source node, 0, to the sink node 3, given the network edge properties and network supply structure. As shown in Table 4.2, there are nine possible flow patterns for the given supply structure, and the goal here is, therefore, to decide which of the flow patterns is optimal by evaluating Equation 4.5. In order to do this, it is required to evaluate the solution quality of each flow pattern: $G(\psi_i)$, $i = 1, \ldots, 9$, which is done by calculating the proportion of feasible states over all network states for each flow pattern. However, due to the huge number of network states, it is impractical to evaluate the exact values by summing over all network states; nevetherless, it is possible to evaluate accurate approximate values using methods introduced in the following chapter (5). Table 4.5 shows the approximate solution qualities for each flow pattern listed in Table 4.2. Given these values, the optimal solution is flow pattern ψ_3.

Flow Pattern	$\approx G(\cdot)$
ψ_1	0.301
ψ_2	0.097
ψ_3	0.372
ψ_4	0.188
ψ_5	0.035
ψ_6	0.252
ψ_7	0.348
ψ_8	0.0999
ψ_9	0.364

Table 4.5: Example STV Network: Flow Patterns with MSE Solution Quality

4.3.3 Problem Complexity: NP-hardness

Pretolani (2000) has shown that finding a simple path with the *a priori* minimum expected travel time in stochastic, time-varying (uncapacitated) networks is NP-hard. This was proved by reduction from the well-known NP-complete Directed Hamiltonian Path Problem (DHPP) (Garey and Johnson, 1979). For time-dependent networks with minimum weighted paths, Orda and Rom (1991) gave a similar proof. Based on these proofs, we show that finding a simple path with the *a priori* maximum MSE value in STV Networks is also NP-hard.

Specifically, by reduction from the DHPP we show that the case where supply $B = 1$ and $b_s(0) = 1$, which is equivalent to the problem of finding a solution $\psi = \{(\sigma, t = 0)\}$, consisting of single directed simple $s - l$ path σ with flow departure time 0, with the maximum MSE value, is NP-hard. Since the case of a single unit of supply forms a sub-class of the more general class, where $B > 1$, the argument extends to the general MSE. Similar to (Pretolani, 2000), we show that this complexity also holds for time-varying networks with deterministic travel times and capacities (the deterministic MSE problem), which again forms a sub-class of the general MSE.

Proof. Consider a directed graph $G = (V, E)$ with two distinguished nodes, $s \in V$ and $l \in V$, $s \neq l$. The goal of the DHPP is to find an $s - l$ path in G that contains all nodes — it is a Hamiltonian path, if it exists. We assume without loss of generality that $(s, l) \notin E$, $\forall i\ (i, s) \notin E$ and $\forall j\ (l, j) \notin E$.

Using G as the topological graph, and setting the peak time horizon $T = |V| - 1$, expanded to $\{0,1,\ldots,T=|V|-1\}$, we define a time-dependent flow network as[3]:

$\forall(s,j)\in E$

$\tau_{sj}(0)=1,\ \rho_{sj}(0)=1$

$\mu_{sj}(0)=1,\ \beta_{sj}(0)=1$

$\forall(i,j)\in E, i\neq s,\ j\neq l$

$\tau_{ij}(t)=1,\ \rho_{ij}(t)=1, \forall t\in\{1,\ldots,T-2\}$

$\mu_{ij}(t)=1,\ \beta_{ij}(t)=1, \forall t\in\{1,\ldots,T-2\}$

$\forall(i,l)\in E$

$\tau_{il}(t)=1, \forall t\in\{1,\ldots,T-1\}$

$\rho_{il}(t)=1, \forall t\in\{1,\ldots,T-1\}$

$$\mu_{il}(t)=\begin{cases}1, & \text{if } t=T-1\\ 0, & \text{if } t<T-1\end{cases}, \forall t\in\{1,\ldots,T-1\}$$

$\beta_{il}(t)=1, \forall t\in\{1,\ldots,T-1\}$

Given this definition, if a unit arrives at a node i, where $(i,l)\in E$, at time $t<T-1$, then $\mu_{il}(t)=0$ and $\beta_{il}(t)=1$ and therefore the solution $\{(\sigma,0)\}$ with $n=|\sigma|$ has an MSE value of 0, it is infeasible:

$$\begin{aligned}\Bigg[\prod_{\substack{1\leq k<n-1\\ i'=i_k,j=i_{k+1}\in\sigma}} P\big(\bar{\mu}_{i'j}(k-1)=1\big)\cdot P\big(\bar{\tau}_{i'j}(k-1)=1\big)\Bigg]\cdot&\\ \Big[P\big(\bar{\mu}_{i_{n-1}i_n}(n-1)=1\big)\cdot P\big(\bar{\tau}_{i_{n-1}i_n}(n-1)=1\big)\Big]&=1\cdot 0\\ &=0\end{aligned}$$

However, if a unit arrives at any node i, $(i,l)\in E$, at $t=T-1$, then $\mu_{il}(t)=1$ and $\beta_{il}(t)=1$, so that the MSE value is

$$\Bigg[\prod_{\substack{1\leq k<n\\ i'=i_k,j=i_{k+1}\in\sigma}} P(\bar{\mu}_{i'j}(k-1)=1)\cdot P(\bar{\tau}_{i'j}(k-1)=1)\Bigg]=1$$

Since paths are assumed simple (see Section 4.1.5), the feasible path will contain $|V|-1$ edges, that is, all nodes in the graph, V, and have the optimal MSE objective function value of 1, if and only if there exists a Hamiltonian $s-l$ path in G. Since only a Hamiltonian $s-l$ path will provide the optimal solution, the MSE is NP-hard. Furthermore, note that the result holds in the size of the network, not a number input like the peak time horizon, T, or the number of edge values, D and H. □

[3]For convenience, since edge properties are defined to happen almost surely (deterministic), property value indices are dropped.

These results indicate that it is unlikely exact and efficient algorithms will be developed to solve the MSE in general, unless P=NP. Note they also indicate that it is unlikely that exact, efficient algorithms will be proposed for the solving the problem in deterministic, time-varying networks either.

4.3.4 Problem Combinatorics

In this section, we explore the combinatorics of the solution space. Firstly, we discuss the combinatorics of the path-space, which underlies the general problem solutions, flow patterns; and, secondly, we discuss bounds on the number of flow patterns.

4.3.4.1 Paths

Solutions to the MSE problem in STV Networks are flow patterns that provide routing instructions for flow to be shipped at specific departure times from the network source to the sink. Flow patterns are composed of paths and departure times and, thus, the number of paths in a network, $|\Omega|$, is an important combinatorial consideration. Note that the feasibility of paths given network properties and problem constraints is not considered here.

Given the network assumptions, the number of paths in a network when the underlying topological digraph is complete[4] provides an upper bound on the number of paths in a graph with nodes $v = |\mathscr{V}| > 2$. Here a complete graph is defined such that

$$\begin{gathered}\forall i,j \in \mathscr{V} \backslash \{s,l\}, i \neq j : (i,j),(j,i) \in \mathscr{E} \\ \forall i \in \mathscr{V} \backslash \{s,l\} : (s,i) \in \mathscr{E} \\ \forall i \in \mathscr{V} \backslash \{s,l\} : (i,l) \in \mathscr{E}\end{gathered}$$

Paths are sequences of distinct nodes from $\mathscr{V} \backslash \{s,l\}$ starting with node s and terminating with node l, so that it is possible to disregard both s and l here. Let $\hat{v} = |\mathscr{V} \backslash \{s,l\}| = v-2$. Given the constraints on paths, as described in the preliminaries in Section 4.1.5, the number of nodes in the paths under consideration is > 2 with length ≥ 2. Then, given the complete topological graph, the number of $s-l$ paths of length $k+1$, where $1 \leq k \leq \hat{v}$, is the k-permutations of $\hat{v}$[5]:

$$P_{\hat{v},k} = \frac{\hat{v}!}{(\hat{v}-k)!}$$

[4]A complete graph is a simple graph, it has no loops or multiedges, with unique oriented edges between all distinct nodes.

[5]This is also known as the falling factorial.

The total number of paths is then the sum over all k-permutations of $\hat{v}$, $k = 1, \ldots, \hat{v} = v - 2$:

$$|\Omega| = \sum_{k=1}^{\hat{v}} P_{\hat{v},k}$$

Finally, the total number of paths grows as a factorial and is bound $O(\hat{v} \cdot \hat{v}!)$.

Proof of the bound:

$$\begin{aligned}
|\Omega| &= \sum_{k=1}^{\hat{v}} P_{\hat{v},k} && (4.6)\\
&= \frac{\hat{v}!}{(\hat{v}-1)!} + \frac{\hat{v}!}{(\hat{v}-2)!} + \cdots + \frac{\hat{v}!}{(\hat{v}-(k=\hat{v}))! = 0! = 1} && (4.7)\\
&= \hat{v} + (\hat{v} \cdot (\hat{v}-1)) + \cdots + \hat{v}! && (4.8)\\
&= O(\hat{v}!) + O(\hat{v}!) + \cdots + O(\hat{v}!) && (4.9)\\
&= O(\hat{v} \cdot \hat{v}!) && (4.10)
\end{aligned}$$

4.3.4.2 Flow Patterns

This section explores the combinatorics of solutions to the MSE problem. Flow patterns are multisets because units of flow can be allocated to the same pairs of paths and departure times. However, across departure times flows departing the source may evaluate differently due to the time-varying nature of network edge properties. Hence, at a particular departure time, say t, we have a multiset for flow $b_s(t)$; but across departure times, order does matter and, hence, overall, for all flow patterns, we have ordered multisets, or permutations of sets of combinations with replacement (multisets). The total number of flow patterns denoted by $|\Psi|$ for network $\mathcal{N}$ is defined by the number of paths in the network, $|\Omega|$, and the availability of network supply, $B = \sum_{t \in S} b_s(t)$.

For each $t \in S$, there are $\left(\!\binom{|\Omega|}{b_s(t)}\!\right)$ possible flow patterns ignoring problem constraints, where $\left(\!\binom{n}{k}\!\right)$ is the multiset coefficient, such that,

$$\begin{aligned}
\left(\!\!\binom{|\Omega|}{b_s(t)}\!\!\right) &= \binom{|\Omega| + b_s(t) - 1}{b_s(t)}\\
&= \binom{|\Omega| + b_s(t) - 1}{|\Omega| - 1}\\
&= \frac{(|\Omega| + b_s(t) - 1)!}{b_s(t)!(|\Omega| - 1)!}
\end{aligned}$$

Then across departure times with available supply, $t \in S$, the number of unique flow patterns has upper bound:

$$|\Psi| \leq \prod_{t \in S} \left(\!\!\binom{|\Omega|}{b_s(t)}\!\!\right) \tag{4.11}$$

This follows by the *rule of product*, where it provides an upper bound since the assumption that the network properties are different across departure times does not necessarily hold.

The maximum number of flow patterns for a given supply vector, with total supply B, is bound:

$$\left(\!\!\binom{|\Omega|}{B}\!\!\right) \leq |\Psi| \leq \prod_{t\in S} \left(\!\!\binom{|\Omega|}{b_s(t)}\!\!\right) \tag{4.12}$$

Assuming network properties are different across departure times, then the r.h.s. provides an upper bound on the number of unique flow patterns, as stated previously. However, if this assumption does not hold, then the l.h.s. provides a lower bound on the maximum number of unique flow patterns. The bounds, however, do not consider the network properties of specific problem instances, and therefore, may severely overestimate the number of candidate solutions.

Section 4.3 defined a novel stochastic optimisation problem, the MSE problem, that attempts to capture the uncertain and transient conditions of emergency movement in the built environment. The problem and its deterministic counterpart were proved to be NP-hard, and the combinatorics of the solution space were discussed. Next, due to the general complexity of the MSE, the following sections develop a method to stochastically bound the optimal solution for the MSE problem.

4.4 Stochastic Bounds on Optimal Solutions

No known algorithms are available to find optimal solutions for the MSE and, in general, given the above complexity results, an efficient and exact algorithm is unlikely to be found. However, in these sections a method for providing stochastic bounds for optimal solutions is developed that can be used to gauge the performance of solution approaches. The bound is in general optimistic, that is, it is in general higher than the optimal value, but it still proves useful for evaluating the proposed solution methods and providing an indicator of the complexity of problem classes.

4.4.1 Interchange Relaxation

The following bounding approach is based on the method detailed in (Mak et al., 1999) for approximating solutions in stochastic programs, and in (Norkin et al., 1998; Gutjahr et al., 1999, 2000) applied in Stochastic Branch-and-Bound approaches. It is an application of the "wait-and-see" bound of Mandansky (1960).

Given that the MSE is a maximisation problem, we are primarily interested in upper bounds on optimal solutions. A lower bound, albeit a potentially poor one, can be evaluated simply using, for example, an interval estimate of the quality of any p-feasible point $\psi \in \Psi, G(\psi) > 0$; or perhaps a more useful one can be found by, for example, a naïve random search algorithm (see Chapter 6). A procedure for determining a stochastic upper bound is defined as follows.

Given the objective in the form of Equation 4.5, by interchanging the maximisation and expectation operators, we can see that

$$z^* = \max_{\psi \in \Psi} \mathbb{E}[h(\psi, \bar{\xi})] \leq \mathbb{E}\big[\max_{\psi \in \Psi} h(\psi, \bar{\xi})\big] \tag{4.13}$$

Therefore, a valid deterministic upper bound for the optimal value is given by

$$U \doteq \mathbb{E}\big[\max_{\psi \in \Psi} h(\psi, \bar{\xi})\big] \tag{4.14}$$

$$= \sum_{\xi \in \Xi} \left[\max_{\psi \in \Psi} h(\psi, \xi)\right] \cdot \mathbb{P}(\xi) \tag{4.15}$$

A stochastic upper bound can thus be obtained using well-known Monte Carlo methods[6]. An unbiased MC estimator for U is given by

$$U_n \doteq \frac{1}{n} \sum_{i=1}^{n} \big[\max_{\psi \in \Psi} h(\psi, \xi^i)\big] \tag{4.16}$$

where $\xi^i, i = 1, \ldots, n$ are independent and identically distributed (i.i.d.) observations from $\bar{\xi}$, that is, independent observations from Ξ according to $\mathbb{P}$. It is unbiased since $\mathbb{E}[U_n] = U$. By the strong law of large numbers, we have asymptotic convergence $U_n \overset{a.s.}{\rightarrow} U$ as $n \rightarrow \infty$.

In general, it is hard to analytically discern the quality of the exact stochastic bound (Equation 4.14). By definition, the optimal solution will be the best across the network state distribution, and the core assumption of the bound is that the better the optimal solution is across the distribution, that is, per network state, the closer the bound. It is therefore in general an optimistic upper bound. Potential improvements to the exact and estimated bounds are discussed in the Future Work section (8.4).

The bounding procedure requires single network states to be generated i.i.d. from the network state distribution. Algorithm 5.2.1 described in Section 5.2.3 provides a procedure for simple random sampling of network states.

[6]For a classic overview of Monte Carlo methods, see Hammersley and Handscomb (1964).

4.4.2 The Deterministic MSE Problem

In order to provide a stochastic upper bound for the MSE, it is necessary to solve a number of deterministic problems, that is, to find feasible flows for single network states. However, in general finding the optimal solution for a single network state is an NP-hard problem, as shown by the complexity results above (Section 4.3.3).

Each network state $\xi \in \Xi$ presents a Deterministic MSE (DMSE) problem, and solving a state ξ to optimality:

$$\max_{\psi \in \Psi} h(\psi, \xi)$$

is, in general, a hard problem. However, since it is not required to provide routing plans and because $\max h(\psi, \xi) = \max I_{\mathcal{F}_\psi}(\xi) = 1$, $\forall \xi \in \Xi$, any feasible flow for the scenario will provide the maximum value: 1. For a sampled set of deterministic problems (network states), the goal thus reduces to finding a feasible flow for each scenario, if one exists, or enumerating the solution space of each problem to show that no feasible flow exists. Note that solving one DMSE problem ensures that there exists at least one p-feasible solution, say $\psi \in \Psi$, where $G(\psi) > 0$, to the general MSE.

Define the set of all feasible network states for all flow patterns $\mathcal{F}_\Psi \subseteq \Xi$, where

$$\mathcal{F}_\Psi = \bigcup_{\psi \in \Psi} \mathcal{F}_\psi$$

and define the indicator function:

$$I_{\mathcal{F}_\Psi}(\xi) = \begin{cases} 1 & \xi \in \mathcal{F}_\Psi \\ 0 & \xi \notin \mathcal{F}_\Psi \end{cases}$$

It is now possible to redefine the optimisation problem such that the bound is

$$\begin{aligned} z^* &= \max_{\psi \in \Psi} \mathbb{E}[h(\psi, \bar{\xi})] \\ &\leq \mathbb{E}\left[I_{\mathcal{F}_\Psi}(\xi)\right] \doteq \hat{U} \end{aligned}$$

and where $\hat{U}$ can be approximated by the unbiased MC estimator

$$\hat{U}_n = \frac{1}{n} \sum_{i=1}^{n} I_{\mathcal{F}_\Psi}(\xi^i) \tag{4.17}$$

and ξ^i, $i = 1, \ldots, n$ are i.i.d. samples from Ξ given $\mathbb{P}$. As before, by the strong law of large numbers, we have convergence $P(\lim_{n \to \infty} \hat{U}_n \to \hat{U}) = 1$.

In general, as previously discussed, solving a deterministic problem (a single network state) is also an NP-hard problem, and therefore it is unlikely that general efficient solution approaches will be proposed. Given the problem's assumptions, no known exact or heuristic algorithms are known for solving the DMSE; so for solving each deterministic problem, we now propose a combined heuristic and greedy procedure.

4.4.3 Solving the Deterministic MSE Problem

The key to an informative upper bound is to be able to efficiently prove that no solution exists for a particular deterministic problem. If no feasible flow exists, then the network state, assuming the source and sink are connected by at least one path, must be over-constrained. However, to prove this requires either a complete enumeration of the space, or a subset of the problem constraints (a proof) that shows that no feasible flow is possible and an efficient way to check that this is the case. Due to the impracticality of using complete enumeration for even reasonably sized problem instances, and the importance of proving that states are infeasible for the bound, several simple heuristics have been developed to check for network properties that lead to infeasibility. A combined heuristic and greedy approach is proposed for tackling deterministic MSE problems, and while this combined approach is not exhaustive, that is, it is not guaranteed to decide if a solution exists or not, empirical results described in Chapter 7 suggest that for the investigation described herein indecision about a state is uncommon (see Section 7.4.4 for more details).

The combined approach involves, firstly, applying two heuristics to determine whether certain conditions arise that lead to infeasibility, and then, secondly, two greedy algorithms that attempt to find a solution to a particular deterministic problem instance. Note that the first heuristic of the two is applied to a MSE problem instance not a particular deterministic sub-problem; whereas the second heuristic is applied to each deterministic MSE problem. Should the heuristics fail, then a greedy algorithm is applied to attempt to find a solution for a particular network state, and should the first greedy approach fail then a variant of the greedy approach is tried. In the case that all the methods fail to decide whether a solution exists or not, the state is discarded and excluded from the evaluation of the bound value.

We now discuss the heuristics and then present the greedy method.

4.4.3.1 Heuristics

Two heuristics are used to prove that no feasible solution exists for a state. This is important since proving that no solution exists is crucial in constructing an informative bound.

The first heuristic is applied to a problem instance to check whether a certain case arises whereby there is no feasible solution to the problem. This is done by inspecting the capacities of the source node out-edges for departure times where supply is available, $t \in S$. If the sum of the maximum possible capacities for each out-edge for a particular departure time, t, is less than the available supply:

$$\sum_{i \in \Gamma^{+1}(s)} \max_{d} \mu_{si}^{d}(t) < b_s(t)$$

then no feasible flow is possible for any network realisation. In other words, for this problem instance the flow conservation constraints will never be satisfied, since no network state can be realised where all supply can be feasibly conveyed from the source to the sink. In this case, the bound is 0.

The second heuristic is applied to each randomly sampled network state rather than the MSE problem instance. The heuristic is similar to the first one, except that it is applied to individual states where edge properties are now realised and hence deterministic. For a sampled state, ξ, similar to before, the out-edges of the source for each departure time, $t \in S$, are inspected and the sum of the realised capacity values: $\sum_{i \in \Gamma^{+1}(s)} \mu_{si}^{\xi}(t)$, is checked to see whether it can support the available supply at the source, $b_s(t)$. Additionally, for each of neighbouring nodes of the source, $i \in \Gamma^{+1}(s)$, the out-edges of i, $j \in \Gamma^{+1}(i)$, are checked to determine whether these edges can also support the available flow:

$$\sum_{i \in \Gamma^{+1}(s)} \min\left(\mu_{si}^{\xi}(t), \sum_{j \in \Gamma^{+1}(i)} \mu_{ij}^{\xi}(t + \tau_{si}^{\xi}(t))\right) < b_s(t)$$

If the case arises then, as before, the flow conservation constraints cannot be satisfied and the bound value for this state is also 0.

If these simple heuristics fail to show that no feasible flow exists, then two greedy approaches, defined next, are used to attempt to find a feasible solution.

4.4.3.2 Greedy Approach

We now propose a greedy algorithm for finding a feasible flow for a given DMSE problem instance. While in general this approach cannot prove that no solution exists, except in the case that not a single feasible path exists, it can prove that a solution does exist.

The greedy approach comprises of a best-first search algorithm for path-finding with a top-level iterative procedure for guaranteeing that all available flow is shipped on feasible paths. The FeasiblePathFinder (FPF) subroutine is based on a best-first search approach (Russell and Norvig, 2003) and works by recursively exploring nodes and building a feasible path from the source node to the sink node in the network given a realised network state. The choice of nodes to explore is based on a best-first criterion. Here, two greedy approaches are defined depending on the best-first criterion: either choose the node with the out-edge that has the minimum amount of feasible flow available or the maximum.

Algorithm 4.4.1 describes a high-level procedure implementing the FPF algorithm. The best-first criterion is implemented using a priority queue, PQ, so that the first element of the queue will the best according to the chosen criterion. To maintain the current state of flow, we define the residual capacity, $\theta_{ij}(t)$, of an edge (i,j) at departure time t, as the current amount of available capacity given the current flow: $\theta_{ij}(t) = \mu^{\xi}_{ij}(t) - x_{ij}(t)$. The algorithm returns a value 1 if a feasible path exists from s to l given the current residual capacity (defined below); otherwise a 0 is returned.

In the worst-case, the FPF algorithm will have to enumerate all simple paths in a general graph. For the complete graph, as defined before, the algorithm would have to enumerate $O(v \cdot v!)$ paths. Nevertheless, in practice, for the problem classes in this investigation (see Chapter 7 for details), the number of edges, and hence number of paths, in the graphs is far less than that of the complete graph.

The top-level procedure, the GreedyFeasibleFlowAlgorithm (GFFA), incrementally builds up a feasible solution using the FPF subroutine to find feasible paths given the current flow. A complete solution is maintained at the top-level so that non-negativity and capacity constraints are maintained feasible, but flow conservation constraints are violated since not all flow has yet arrived at the sink, and hence the network's demand is unsatisfied. The algorithm stops once all constraints are satisfied, that is, once a flow pattern has been found that conveys all available flow for each $t \in S$ from the source to the sink on feasible paths. If at any point the subroutine fails to find a feasible path, then the search process terminates and then either the second greedy approach is executed or the state is discarded from the bound.

Algorithm 4.4.1: FEASIBLEPATHFINDER(i, l, t, σ, x)

```
comment: Add out-edges to queue, prioritised according to best-first criterion.
$PQ.insert(j, \theta_{ij}(t)), \forall j \in \Gamma^{+1}(i)$
while $PQ \neq \emptyset$
  do {
    $j \leftarrow PQ.firstElement()$
    if $j \in \sigma$
      then continue
    $x' \leftarrow \min(x, \theta_{ij}(t))$
    $t' \leftarrow t + \tau_{ij}(t)$
    if $x' = 0$ or $t' > T$
      then continue
    $\sigma \leftarrow \sigma \cup \{j\}$
    if $j = l$
      then return $(x')$
      else $\hat{x} \leftarrow$ FeasiblePathFinder$(j, l, t', \sigma, x')$
    if $\hat{x} > 0$
      then return $(\hat{x})$
      else $\sigma \leftarrow \sigma \backslash \{j\}$
return $(0)$
```

Algorithm 4.4.2 describes the GFFA approach. We define an excess, $e(t)$, for each $t \in [T]$, which specifies the amount of flow remaining to be shipped from the source at each time interval given the current flow. If the algorithm finds a feasible flow for the network state, it returns a 1; otherwise a 0 is returned.

Algorithm 4.4.2: GreedyFeasibleFlowAlgorithm($\mathcal{N}$, ξ)

```
$x_{ij}(t) \leftarrow 0, \forall (i,j) \in \mathcal{E}, t \in [T]$
$\theta_{ij}(t) \leftarrow \mu_{ij}^{\xi}(t), \forall (i,j) \in \mathcal{E}, t \in [T]$
$e(t) \leftarrow b_s(t), \forall t \in [T]$
while $\{t \in S | e(t) > 0\} \neq \emptyset$
do
    $\hat{t} \leftarrow \min\{t \in S | e(t) > 0\}$
    $\sigma \leftarrow \{s\}$
    $x' \leftarrow$ FeasiblePathFinder$(s, l, \hat{t}, \sigma, \infty)$
    $\varepsilon \leftarrow \min(e(\hat{t}), x')$
    if $\varepsilon = 0$
      then return $(0)$
    $e(\hat{t}) \leftarrow e(\hat{t}) - \varepsilon$
    comment: Update flow matrix:
    $x_{i_k i_{k+1}}(t') \leftarrow x_{i_k i_{k+1}}(t') + \varepsilon, \forall i_k \in \sigma, 1 \leq k < |\sigma|, t'_{k,k+1} = t'_{k-1,k} + \tau_{k-1,k}^{\xi}(t'_{k-1,k})$
    comment: Update residual capacities:
    $\theta_{ij}(t') \leftarrow \mu_{ij}^{\xi}(t') - x_{ij}(t'), \forall (i,j) \in \mathcal{E}, t' \in [T]$
return $(1)$
```

The GFFA has worst-case computational complexity $O(B \cdot F)$, where F is the complexity of the path-finding algorithm. The maximum number of iterations for the GFFA is equal to the total supply, B, and since the GBFS has a worst-case complexity $O(v \cdot v!)$, the algorithm's worst-case complexity is $O(B \cdot (v \cdot v!))$. The complexity results indicate that in the worst-case this approach is impractical, even for finding a single feasible path. Nevertheless, for the work described herein, the graphs have a low connectivity and hence significantly less paths than the worst-case. Furthermore, the empirical results described in Chapter 7 suggest that for the networks of interest here it is practical and efficient to evaluate the bound.

4.5 Summary

In this chapter, a novel stochastic network flow problem, the Maximal Safest Escape problem, was formally defined that captures the uncertainty and transiency of conditions during emergency movement in the built environment. Given a network, the goal of the problem is to find the a priori flow patterns with the highest probability of successfully conveying the flow in the network from the source to the sink. In other words, the MSE problem models the task of finding the set of paths with the highest likelihood of getting building occupants from their starting locations to places of safety. The problem is formalised in STV Networks, and shown to be NP-hard. Additionally, a stochastic bounding procedure is developed to provide upper bounds to optimal solutions for gauging the performance of the proposed metaheuristic-based solution approaches.

Chapter 5

Candidate Solution Generation, Evaluation and Ranking

This chapter proposes methods for generating, evaluating and ranking candidate solutions to the MSE. These methods are required by the herein proposed solution approaches (detailed in the following chapter (6)). In general, generating p-feasible solutions is a hard problem, and so a simple method for generating approximately equally likely solutions is presented. Furthermore, in general, the evaluation of the exact quality of solutions is impractical due to the uncertainty in the network, and hence a simulation approximation method for estimating solution quality is given. In noisy domains, the comparison of solutions with finite resources is also uncertain, and therefore the approximation method is integrated into a statistical ranking procedure from the literature, in order to guarantee to a predefined confidence level that the comparing of solutions based on their estimated quality is correct.

5.1 Candidate Solution Generation

A candidate solution to the MSE is a flow pattern, represented by a multiset, consisting of pairs of paths and departure times with a multiplicity function that determines the number of units of flow assigned to each path and departure time. The proposed solution approaches require methods for generating candidate solutions, since they are based on the well-known generate-and-test paradigm. However, the problem of generating feasible or reasonable quality solutions for hard (S)COPs is often a hard problem too; in fact, for the MSE, in the worst case it can involve solving the optimisation problem itself, thus also making it an NP-hard problem. This issue is exacerbated in stochastic environments where exact solution quality is often unavailable (as is discussed in the following section (5.2)). For the MSE, in general, generating p-feasible solutions is a hard problem because accounting for feasibility would require evaluating solutions; therefore, here candidate solutions are generated without direct consideration for feasibility.

5.1.1 Random Paths and Flows in STV Networks

The herein described solution approaches require the generation of candidate solutions. In order to generate uniformly random flow patterns a simple procedure is defined so that all solutions, $\psi \in \Psi$, are equally likely to be generated. The method works iteratively, finding approximately equally likely paths in the topological graph for each of the B units of flow. In the case that more than one unit is allocated to the same path and departure time, these units of flow are summed and allocated to a single path and departure time pair.

Underlying this procedure is a simple method for finding random simple $s-l$ paths in the topological graph $\mathscr{G}$ of the network $\mathscr{N}$. However, due to a potential bias in the selection of paths on the topological graph, that is, that the selection of every path is not equiprobable due to the structure of the underlying graph, a method for finding approximately uniformly likely flow patterns for a given supply is presented. To highlight the potential bias the problem is first demonstrated for directed acyclic graphs and then for directed general graphs.

5.1.2 Directed Acyclic Networks

An example of such a bias can be seen in Figure 5.1. The graph contains three $0-5$ paths: $(0,1,3,5)$, $(0,1,4,5)$ and $(0,2,5)$. Note that node 1 is incident on two of the three paths. Hence, assuming that a graph traversal method, such as depth-first search, is used for finding paths, to allow for the equiprobable selection of each path, node 1 should be selected twice out of every three path searches as the successor node of node 0. So, the probability of selecting node 1 from node 0, $p_0(1) = 2/3$, whereas for node 2, it is $p_0(2) = 1/3$. The transition matrix for the graph in Figure 5.1 is given by the matrix $Q = [q_{ij}]_{v\times v}$, where Q has size v^2. A $q_{ij} > 0$ entry means $(i,j) \in \mathscr{E}$; however, $q_{ij} = 0$ can either mean $(i,j) \notin \mathscr{E}$, or that no $s-l$ paths exist where (i,j) is incident. Then, the probability of choosing j from i, $p_i(j)$, is given by the entry q_{ij}; and, as required, we have that $\sum_{j=1}^{v} q_{ij} = 1$, for each row $i = 1,\dots,v$. Also, since no waiting is allowed, $q_{ii} = 0, \forall i$, and since $\Gamma^{+1}(l) = \{\}$, $q_{l\cdot} = 0$.

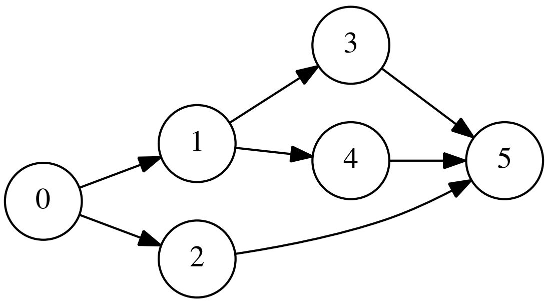


Figure 5.1: Example graph

$$Q = \begin{pmatrix} 0 & 2/3 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Given a topological graph, random simple paths are generated by a randomised depth-first search procedure on the graph[1]. The randomised depth-first search procedure expands to the next node based on a given probability distribution. A simple example would be if all nodes were equally likely, modelled by a uniform distribution, so that each neighbouring node, $j \in \Gamma^{+1}(i)$, would have equal probability of being selected from node i: $p_i(j) = \frac{1}{|\Gamma^{+1}(i)|}$.

As demonstrated by the above example, to allow the uniformly likely exploration of the path-space, the probability of choosing an adjacent node, say j from node i, should be weighted by the number of $(i, j, \ldots, l)$ paths in the graph. The set of $s-l$ paths in graph $\mathscr{G}$ is denoted Ω (see Notation and Preliminaries (4.1.5)). A subset of this set from a node, say i, to l, denoted $\Omega_i \subseteq \Omega$, is the set of $(s, \ldots, i, \ldots, l)$ paths in $\mathscr{G}$. Now to ensure a uniform selection of paths, the probability of choosing a successor node $j \in \Gamma^{+1}(i)$ from node i, $p_i(j)$, is given by the number of paths where $(i, j, \ldots, l) =$

[1]For a description of a generic depth-first procedure, see, for example, (Russell and Norvig, 2003).

$\{\sigma \in \Omega_i | i_k, j_{k+1} \in \sigma\} = \Omega_{(i,j)} \subseteq \Omega_i$, over all paths where i is incident, so that $p_i(j) = \frac{\Omega_{(i,j)}}{\Omega_i = \sum_{j' \in \Gamma^{+1}(i)} \Omega_{(i,j')}}$, and $\sum_{j \in \Gamma^{+1}(i)} p_i(j) = 1$, $\forall i \in \mathscr{V} \backslash \{l\}$. These distributions are then used by a weighted depth-first search procedure for selecting successor nodes, rather than, for example, employing a uniform choice selector of successor nodes, i.e. $p_i(j) = \frac{1}{|\Gamma^{+1}(i)|}, \forall j \in \Gamma^{+1}(i), \forall i \in \mathscr{V} \backslash \{l\}$.

5.1.3 Directed Cyclic Networks

As shown above, in theory a simple way to counter the bias in the graph is to count the number of paths in the graph, and then to weight the choice of successor node accordingly. However, the problem of counting simple $s-l$ paths in directed general graphs is known to be #P-complete[2] (Valiant, 1979), and therefore it is highly unlikely that efficient algorithms will be found for addressing this problem in general. To cope with this, a method is proposed here for estimating the transition matrix in order to generate approximately equally likely paths. The method is as follows.

A set of $s-l$ paths $\hat{\Omega} = \{\sigma_k\}_{k=1}^n$ are i.i.d. generated by a random depth-first procedure, where $p_i(j) = \frac{1}{|\Gamma^{+1}(i)|}, \forall j \in \Gamma^{+1}(i), \forall i \in \mathscr{V} \backslash \{l\}$. For each node $i \in \mathscr{V} \backslash \{l\}$, we have, $\hat{\Omega}_i$, the set of paths from $s-l$ where node i is incident, and for each $j \in \Gamma^{+1}(i)$, for all $i \in \mathscr{V} \backslash \{l\}$, we have $\hat{\Omega}_{(i,j)}$, the set of all paths from node i to the sink l in which the edge (i,j) is incident.

Define the transition matrix, $\hat{Q}_n = [\hat{q}_{ij}]_{v \times v}$, with initial entries[3] $\hat{q}_{ij} = 1, \forall (i,j) \in \mathscr{E}$. Then for each (i,j) where $j \in \Gamma^{+1}(i)$, for all $i \in \mathscr{V} \backslash \{l\}$, we set the value $\hat{q}_{ij} = |\hat{\Omega}_{(i,j)}|$. On termination, the matrix $\hat{Q}_n$ provides an approximation of the true transition matrix Q, and as n increases and more of the path-space is discovered, $\hat{Q}_n \to Q$. The matrix $\hat{Q}_n$ is then used for weighting the random selection of nodes in order to generate approximately uniformly random paths (and hence solutions) — where the probability of choosing a node $j \in \Gamma^{+1}(i)$ from i is $p_i(j) = \frac{\hat{q}_{ij}}{\sum_{j' \in \Gamma^{+1}(i)} \hat{q}_{ij'}}$.

This procedure is executed once per problem instance. To improve the efficiency of the process in practice, a single path search is stopped if too much backtracking

[2]The computational complexity class #P can be viewed as the counting version of the class NP, that is, it is the set of enumeration or counting problems that can be solved by a non-deterministic algorithm running in polynomial time (Garey and Johnson, 1979). Those counting problems that are #P-complete are, analogously to NP and NP-complete, those problems in #P where all other problems in #P can be polynomially reduced to them; they are the hardest problems in #P. Examples of counting problems include finding the number of satisfying truth assignments for an instance of the Satisfiability problem, or counting the number of distinct Hamiltonian circuits in a graph. For more details, see, for example, (Garey and Johnson, 1979).

[3]The value is set at 1 so that no edge has a probability of 0 of getting selected.

occurs, and the search is restarted. Pilot studies were conducted, and it was established that for a single path, a reasonable threshold for the number of node visits is 1000000, and 10000 is reasonable for the total number of paths searches per instance.

Paths are then generated using a random depth-first procedure with the matrix $\hat{Q}_n$ used for weighted sampling without replacement of the order of nodes to explore, so that all paths are approximately equally likely to be found. Flow patterns can then be generated by assigning a random path to each unit of flow $b_s(t)$, $\forall t \in S$, where $S = \{t \in [T] | b_s(t) > 0\}$ and $\sum_{t \in S} b_s(t) = B$. Since all paths are (approximately) likely to be generated and assigned individually to each unit of flow, all flow patterns are approximately likely to be generated. In the case that identical paths are generated for the same departure time, the multiplicity value for the appropriate pair of the solution is incremented as required.

5.2 Candidate Solution Evaluation

In stochastic environments with problem objectives that involve expectations it is often the case that the exact evaluation of the quality of potential solutions is impractical. The complexity arises due to the need to evaluate expectations over large joint distributions of random variables, of which the dimensionality can be very high. Evaluating the MSE solution quality of candidate solutions (Equation 4.4), in general, involves summations over potentially high dimensional spaces, where the number of network states grows exponentially. Therefore, in the general case, it is impractical to evaluate exact solution quality, using, for example, dynamic programming techniques, and, in practice, it is common to employ approximations instead.

The literature on metaheuristics for SCOPs (as discussed in Section 3.5) suggests two general approaches for evaluating the approximate solution quality of solutions, assuming closed-form expressions are either not available or impractical to evaluate. The first is ad-hoc and fast, but problem-specific, approximations, and the second, simulation approximation using, for example, Monte-Carlo (MC) sampling methods. Due to the simplicity and efficiency of simulation approximation in general, and the absence of a reliable approximation function, MC methods are employed here.

The following sections firstly detail the formulation of the employed simulation approach. Then, secondly, two sampling frames for use with the simulation approach are discussed. This is followed by a brief discussion on the issues of comparing solutions in stochastic domains, including the MSE. Finally, the chosen statistical ranking framework and its integration with the main solution approach is explicated.

5.2.1 Simulation Approximation for Estimating Solution Quality

Due to the complexity of evaluating the exact quality of solutions in general, it is proposed to use a simulation approximation approach to estimate solution quality. A common simulation approximation approach is given by the Monte Carlo sampling method.

The exact solution quality of a solution, $\psi \in \Psi$, for the MSE is given by $G(\psi) = \mathbb{E}[h(\psi, \bar{\xi})]$ (Equation 4.4), with variance[4] denoted ς^2_ψ. Due to the impracticality of enumerating the support of $\bar{\xi}$ in general and hence evaluating the expected value exactly, as discussed at the beginning of the section (5.2), it is proposed to estimate the value using a simulation approximation approach based on MC sampling.

It is possible to estimate $G(\psi)$ using the unbiased MC estimator

$$\hat{G}^n_\psi = \frac{1}{n}\sum_{i=1}^{n} h(\psi, \xi^i) \tag{5.1}$$

where $\xi^1, \ldots, \xi^n$ are i.i.d. states sampled from Ξ, the set of all network states, given $\mathbb{P}$. Since G_ψ is the unknown parameter of a Bernoulli random variable, $\hat{G}^n_\psi$ estimates the proportion of successes (feasible states).

We also have an estimator of ς^2_ψ, the sample variance

$$s^2_{\psi,n} = \hat{G}^n_\psi(1 - \hat{G}^n_\psi) \tag{5.2}$$

Then the strong law of large numbers guarantees that we have asymptotic convergence almost surely, $P(\lim_{n\to\infty} \hat{G}^n_\psi = G_\psi) = 1$.

5.2.1.1 Approximation Issues

There are, however, two well-known issues with this approach. Firstly, the normal approximation for the unknown Bernoulli parameter is known to perform poorly with small sample sizes and when estimating tail values, i.e. values close to 0 or 1. In fact, it is recommended not to use the approximation with small sample sizes; in this work, where small sample sizes are used, a direct estimate of the solution quality is not actually required during search (for more details, see Section 6.1.7 of the following chapter (6)).

[4]Conventionally σ is used to denote population variance; however, here σ denotes paths in the network, and therefore the alternate ς is used to prevent misreadings.

Secondly, the approach is known to have a slow convergence rate to the true mean, $O(1/\sqrt{n})$. A number of methods exist to improve the rate, common ones including so-called variance-reduction techniques[5] that attempt to reduce the variance of the estimates. It is often advisable to employ such techniques as they can substantially improve the rate of convergence of crude MC sampling. They are however not employed here due to the assumptions of the statistical ranking procedure, discussed in Section 5.4.1. In fact, the ranking procedure attempts to avoid the convergence rate issue altogether, by focussing effort not on obtaining accurate estimates of solution quality but on obtaining a level of confidence in the differences in quality among solutions, and hence their ordinal ranking.

5.2.2 Precision of Simulation Approximation

An additional issue concerns the numerical precision of the estimates of the objective of the MSE. In the context of the MSE, these values have a domain-specific interpretation; for example, given environmental conditions, a solution with $G(\psi) = 0.25$, has a 25% chance of successful traversal from a starting location to a place of safety in a building. Therefore, in practice, values < 0.01 could be interpreted as practically insignificant given the problem domain. In practice, this issue effects sample sizes and, therefore, sample precision and the significance of the differences in quality between solutions.

The range of the objective function is $[0,1]$ and hence the largest observable difference between two individuals is $1-0=1$. But what is the smallest observable fitness difference to achieve the desired precision? The precision of the sampling procedure is determined by the number of samples, $n \in \mathbb{N}$, since the minimum MSE value (> 0) is $1/n$, that is, if one of the sampled states is feasible out of n. If we write $n = c \cdot 10^k$, where $c \in \mathbb{N}, k \in \mathbb{N}_0$, then the minimum precision of the MSE value is $\frac{1}{c \cdot 10^k}$. This means that in order to increase the precision by an order of magnitude it is necessary to increment k, which amounts to an increase of samples by an order of magnitude. Table 5.1 provides examples of different sample sizes and the precision they afford. Given the problem and domain, the minimum practically significant value is taken to be 0.01, which represents a 1% chance of reliability; for this precision the minimum number of samples is $n = 100$.

[5]For examples, see Bratley et al. (1987) and Law and Kelton (2000).

Table 5.1: Precision of Sample Sizes

Sample Size	Minimum Precision
1	$1/1 = 1$
10	$1/10 = 0.1$
50	$1/50 = 0.02$
100	$1/100 = 0.01$
250	$1/250 = 0.004$
500	$1/500 = 0.002$
1000	$1/1000 = 0.001$
5000	$1/5000 = 0.0002$
10000	$1/10000 = 0.0001$

5.2.3 Sampling Frame: Network States

In order to implement a simulation approximation approach it is necessary to decide on the underlying distribution from which to sample, the sampling frame. Until now it has been assumed that the frame is the population of network states, Ξ, as proposed by Opasanon (2004) and Miller-Hooks and Sorrel (2008). In this case, as noted previously, the number of network states is $(D \cdot H)^{e(T+1)}$, thus the number of network states grows exponentially in the size of the network, the time horizon, and the sizes of the support for edge property random variables.

The advantages of using the population of network states as the sampling frame include that it is problem independent and it is not a function of the specific solution to be evaluated. For their respective SCOPs, Opasanon (2004) and Miller-Hooks and Sorrel (2008) used similar sampling designs, so that a set of sampled network states was used to evaluate all solutions each iteration of the heuristic algorithm. While using network states provides a relatively simple sampling frame, it however samples many edge property events that, given the assumptions of stochastic independence, may actually have no bearing on the evaluation of individual candidate solutions, unless the solution includes all edges over all time periods. Nevertheless, if evaluating over all states these non-effecting events will sum out; however, the use of a sampling approach would imply that not all states will be enumerated in the evaluation. So, despite the simplicity of using the population of network states, since all random variables for all edges over time are sampled, the dimensionality of the sample space may be unnecessarily high.

Algorithm 5.2.1 describes a high-level procedure for sampling from Ξ given $\mathbb{P}$ by simulating the underlying random variables, since the population of states is not directly accessible. The algorithm samples a travel time and capacity for each edge, $(i, j) \in \mathscr{E}$, and for each time interval, $t \in [T]$. Note that this algorithm is generic for STV Networks since it does not consider the problem objective or constraints.

Algorithm 5.2.1: SampleNetworkState($\mathscr{N}$)

```
comment: Initialisation of state:
$\tau^{\xi}_{ij}(t) \leftarrow -1, \forall (i,j) \in \mathscr{E}, \forall t \in [T]$
$\mu^{\xi}_{ij}(t) \leftarrow -1, \forall (i,j) \in \mathscr{E}, \forall t \in [T]$
comment: Sampling of state:
for each $(i,j) \in \mathscr{E}$
  do for each $t \in [T]$
       do comment: Sample travel time value:
          $\tau^{\xi}_{ij}(t) \leftarrow$ SampleProbabilityDistribution($\tau_{ij}(t), \rho_{ij}(t)$)
          comment: Sample capacity value:
          $\mu^{\xi}_{ij}(t) \leftarrow$ SampleProbabilityDistribution($\mu_{ij}(t), \beta_{ij}(t)$)
return ($\xi$)
```

The outer loop will always enumerate all edges and as such has computational complexity $\Theta(e)$. The inner loop will always enumerate all time intervals, $t \in [T]$, so it has complexity $\Theta(T)$. The sampling of values from distributions in the worst-case will loop in the size of the support of each distribution. Furthermore, in the worst-case, the graph will be fully connected so $e = O(v^2)$, and therefore the algorithm has a worst-case time complexity $O(v^2 T(D+H))$ and best-case complexity $\Omega(eT)$.

The following procedure describes the method used by Algorithm 5.2.1 for randomly sampling values from probability distributions. It uses the inverse transform method[6] such that given a sampled value from a uniform distribution, $u \in (0,1)$, it returns x where $x = F^{-1}(u)$, where $F^{-1}(\cdot)$ is the inverse distribution function. The distribution is input in the form of an indexed set of values (the support of the distribution) and associated probabilities.

[6]For more information see, for example, (Ross, 2007).

Algorithm 5.2.2: SAMPLEPROBABILITYDISTRIBUTION(x, p)

```
sample $u \sim U(0,1)$
$i \leftarrow 1$
$s \leftarrow p_i$
while $s < u$
  do $\begin{cases} i \leftarrow i+1 \\ s \leftarrow s+p_i \end{cases}$
return $(x_i)$
```

In the worst-case the sampling procedure will loop $O(|x|)$ times, where $|x|$ is the size of the support, x.

5.2.4 Sampling Frame: Flow Pattern States

As discussed above, given the stochastic independence assumptions, evaluating candidate solutions on network states may include many edge property random events that do not have any bearing on the feasibility of a solution. Although these values will sum out when evaluating over all states, it is impractical to enumerate all states in general, therefore we are sampling from a potentially much larger distribution than necessary. It is possible to reduce the dimensionality of the sample space, and hence, in general, make estimates more accurate with the same effort.

Now the sample space is a function of a flow pattern, and elements of the space are called flow pattern states. For a flow pattern $\psi \in \Psi$, the random event of a flow pattern state is denoted $\bar{\xi}_\psi$, and the population of flow pattern states by Ξ_ψ. The random events that may actually effect the solution quality of individuals are by definition a subset of the network state events, and hence $|\Xi_\psi| \leq |\Xi|$.

The probability of a particular flow pattern state $\xi_\psi \in \Xi_\psi$ is given by the product of the product of the probabilities of each assignment of capacity and travel time, for all edges in a path, for all paths in a flow pattern.

$$\begin{aligned}\mathbb{P}_2(\bar{\xi}_\psi = \xi_\psi) &= P\Big(\bigcap \bigcup_{\substack{i_k, i_{k+1} \in \sigma, \\ 1 \le k < |\sigma| \\ (\sigma, t_0) \in \psi}} \bar{\mu}_{i_k i_{k+1}}(t_k) = \mu^{\xi_\psi}_{i_k i_{k+1}}(t_k) \cap \bar{\tau}_{i_k i_{k+1}}(t_k) = \tau^{\xi_\psi}_{i_k i_{k+1}}(t_k)\Big) \\ &= \prod \bigcup_{\substack{i_k, i_{k+1} \in \sigma, \\ 1 \le k < |\sigma| \\ (\sigma, t_0) \in \psi}} P(\bar{\mu}_{i_k i_{k+1}}(t_k) = \mu^{\xi_\psi}_{i_k i_{k+1}}(t_k)) \cdot P(\bar{\tau}_{i_k i_{k+1}}(t_k) = \tau^{\xi_\psi}_{i_k i_{k+1}}(t_k))\end{aligned}$$

where $t_k = t_{k-1} + \tau_{k-1k}(t_{k-1})$ for $k > 1$, and $t_1 = t_0$.

A drawback of this approach is that it is now necessary to consider the feasibility of candidates directly because the generation of states is based on a candidate solution and the set of events that have direct impact on the feasibility of the candidate.

Based on this, for a flow pattern state, $\xi_\psi \in \Xi_\psi$, the value of the indicator function is extended to:

$$I_{\mathcal{F}_\psi}(\xi_\psi) = \begin{cases} 1 & \xi_\psi \in \mathcal{F}_\psi \\ 0 & \xi_\psi \notin \mathcal{F}_\psi \end{cases}$$

Then we redefine the problem objective, so that $\forall \psi \in \Psi$:

$$\mathbb{P}_2(\mathcal{F}_\psi) = \sum_{\xi_\psi \in \Xi_\psi} I_{\mathcal{F}_\psi}(\xi_\psi) \cdot \mathbb{P}_2(\xi_\psi) \tag{5.3}$$

$$= \mathbb{E}[I_{\mathcal{F}_\psi}(\bar{\xi}_\psi)] \tag{5.4}$$

That is, the probability of the set of feasible states for flow pattern ψ is equal to the sum of the individual probabilities of each feasible flow pattern state.

Furthermore, under the stochastic assumption of independence, we have that

$$\mathbb{E}[I_{\mathcal{F}_\psi}(\bar{\xi}_\psi)] = \mathbb{E}[h(\psi, \bar{\xi})]$$

By the combinatorics of states, each flow pattern state is a subset of a number of network states. Therefore, in the simplest case, if a flow pattern includes all edges over all time periods, then the set of states are identical, and trivially the evaluation is equal. In the more likely case that they do not include the same events, the values are still the equal because, in the summation of all states, random variables that have no effect on feasibility in the network states are summed out. Despite the reduction in dimensionality, it is still impractical in general to enumerate all flow pattern states; the gain is in the reduction of dimensionality for sampling.

Algorithm 5.2.3 describes a procedure for sampling from the flow pattern state distribution of a given solution $\psi \in \Psi$. Rather than enumerating over all edges and over all time intervals, as Algorithm 5.2.1 does, the algorithm evaluates only those edges and time intervals relevant to a particular flow pattern. In the algorithm the set Z maintains pairs of edges and departure times that represent the current state of the joint flow. The map $\bar{R}_{ijt}$ stores pairs of paths and departure times $(\sigma, t_0) \in R$ associated with the joint flow for edge (i, j) at departure time t. A single loop of the algorithm entails sampling a travel time and capacity value for the current edge and departure time, $((i, j), t) \in Z$, which is determined by the earliest available flow. Ties are broken arbitrarily between which pair to choose for the minimum t. Once edge properties have been sampled, then the current joint flow is advanced in time and new pairs, $((i, j), t)$, are added to the set Z, and $\bar{R}_{ijt}$ is updated accordingly. This cycle continues until no pairs are left, that is, all flow has arrived at the network sink l, or it stops if the sampled state is infeasible. If the sampled state is infeasible, then the procedure returns a 'null' value, otherwise the feasible sampled state, ξ_ψ, is returned.

In the worst-case, the complexity of the algorithm is the same as sampling from the network state distribution. For the set Z, all pairs of edges and time intervals enter at most once, so that in the worst-case this is all edges $(i, j) \in \mathscr{E}$ over time $t \in [T]$, $O(v^2T)$. Again, the worst-case for the sampling of distributions will loop in the size of the support of each distribution. Overall, the algorithm's worst-case computational complexity is $O(v^2T(D+H))$; however, its lower bound may be less than that of Algorithm 5.2.1 since it is dependent on the candidate solution to be evaluated.

Algorithm 5.2.3: SAMPLEFLOWPATTERNSTATE($\mathcal{N}, \psi = (R, m)$)

```
comment: Initialisation of state $\xi_\psi$:
$\tau_{ij}^{\xi_\psi}(t) \leftarrow -1, \forall (i,j) \in \mathcal{E}, \forall t \in [T]$
$\mu_{ij}^{\xi_\psi}(t) \leftarrow -1, \forall (i,j) \in \mathcal{E}, \forall t \in [T]$
comment: Initialisation of data structures:
$\bar{R}_{ijt} \leftarrow \emptyset, \forall (i,j) \in \mathcal{E}, \forall t \in [T]$
$Z \leftarrow \{((i_1,i_2),t_0) | i_1 = s, i_2 \in \sigma, (\sigma,t_0) \in R, m((\sigma,t_0)) > 0\}$
$\bar{R}_{ijt} \leftarrow \{(\sigma,t) \in R | i = i_k, j = i_{k+1} \in \sigma, m((\sigma,t)) > 0\}, \forall ((i,j),t) \in Z$
comment: Sampling of state:
while $Z \neq \emptyset$
  do
    $((i,j),t) \leftarrow \arg\min_{((i',j'),t') \in Z} t'$
    $\tau_{ij}^{\xi_\psi}(t) \leftarrow$ SampleProbabilityDistribution($\tau_{ij}(t), \rho_{ij}(t)$)
    $t' \leftarrow t + \tau_{ij}^{\xi_\psi}(t)$
    if $t' > T$
      then return ( null )
    $\mu_{ij}^{\xi_\psi}(t) \leftarrow$ SampleProbabilityDistribution($\mu_{ij}(t), \beta_{ij}(t)$)
    $\hat{x} \leftarrow \sum_{(\sigma,t_0) \in \bar{R}_{ijt}} m((\sigma,t_0))$
    if $\hat{x} > \mu_{ij}^{\xi_\psi}(t)$
      then return ( null )
    if $j \neq l$
      then
        $Z' \leftarrow \{((j,j'),t') | j = i_k, j' = i_{k+1} \in \sigma, (\sigma,t_0) \in \bar{R}_{ijt}\}$
        $\bar{R}_{jj't'} \leftarrow \bar{R}_{jj't'} \cup \{(\sigma,t_0) \in \bar{R}_{ijt} | j = i_k, j' = i_{k+1} \in \sigma\}, \forall ((j,j'),t') \in Z'$
        $Z \leftarrow Z \cup Z'$
    $Z \leftarrow Z \backslash \{((i,j),t)\}$
return ($\xi_\psi$)
```

5.3 Comparing Candidate Solutions

The preceding sections proposed an efficient way of approximating the quality of individual solutions, that is, a particular flow pattern applied to the given network. However, in the context of optimisation the problem objective is to find the best solution in the solution space, and thus it is necessary to be able to discriminate between solutions. In deterministic environments, evaluating the quality of solutions is often efficient and certain, and hence comparisons between candidate solutions are often efficient and certain too. In stochastic domains, due to the complexity and uncertainty in evaluating the exact quality of individuals, this is often not the case, and it is common that the balance of effort increasingly shifts from the exploration of the search space to the evaluation of solutions and the exploration of the search space. Comparing individuals in stochastic environments with finite resources introduces the possibility of selection error, where uncertainty and approximation result in the wrong solution chosen in a comparison (e.g. of two solutions, the solution with the actual worse performance is chosen). In cases where statistical sampling is used, as suggested above, an estimation error is introduced due to the randomisation of the evaluation, and hence with finite resources comparisons between solutions are executed with some probability of error. To control this, statistically based techniques have been devised to guarantee to a certain confidence level that the comparisons are in fact correct.

Since for the MSE it is not known whether the variance of each solution's evaluation will vary across the space, and it is, therefore, potentially sub-optimal, or misleading, to allocate the same number of samples to each individual; for example, by using an a priori fixed resampling rate. The most efficient sampling design would allocate the minimum number of samples required to discriminate between solutions with some level of confidence, so that the algorithm is still able to find the optimal solution. Of course, the effort required to distinguish between individuals will vary depending on the size and variance of the difference in their qualities (the difference is also a random variable). In particular, distinguishing between solutions that have similar qualities or a high variance requires more effort than those with a large difference or that have a low variance.

As discussed in the literature review (in Sections 3.4 and 3.5) a number of theoretical and practical approaches have been suggested for tackling problems in stochastic/noisy environments using EAs. In particular, the review highlighted a trend towards combining metaheuristics with statistical ranking and selection procedures, and noted how effective this symbiosis can be. Based on this, as previously discussed, a recently proposed framework by Schmidt et al. (2006) and Schmidt (2007) that tightly integrates EAs with a state-of-the-art ranking and selection (R&S) procedure, the OCBA, is applied here. Building on its introduction in Section 3.4.3, the following sections describe in detail the original framework, and several new extensions: an optimisation and a heuristic procedure for speeding up each stage of the OCBA, and the use of informative priors for Bayesian inference over multiple selection problems.

5.4 Statistical Selection and Ranking

In the stochastic/noisy context described herein, applying R&S procedures may be a fruitful approach to handling the uncertainty, as discussed in Section 3.5. In Simulation Optimisation (SO), where R&S methods are often applied, they are often used to choose the best, or a subset of the best, simulation configuration from a set of finite alternatives given some criteria, e.g. the expected performance[7]. In SO, the performance of simulation systems is uncertain and as such are modelled by random variables. The goal is then to pick the best configuration with some probability of making the correct selection, e.g. to pick the best from k systems with probability 95%. The core idea here, taken from Ordinal Optimisation, is that *ordinal* ranking of systems, that is, ranking based on establishing which system is first, then second, then third, etc., is easier than ranking based on accurate estimates of system performance (Ho et al., 1992). Dai (1996) proved exponential convergence to the correct ordering under certain conditions for ordinal comparisons.

For sequential procedures, the general approach to the selection-of-the-best problem is as follows. Firstly, simulate each system a number of times and evaluate the performance of each system using statistics on the evidence (the data generated). Secondly, use statistics to compare the systems to determine the current best based on the observed performances. Finally, sequentially allocate additional simulations and compare the systems again, until some goal, defined by stopping criteria, is achieved, e.g. the selection of the best system to at least some confidence level. In these contexts, the event of determining the true ordering is termed the ‘correct selection’, with the likelihood of having found the actual best system the ‘probability of correct selection’ (PCS). When an indifference-zone is used this likelihood is termed the ‘probability of good selection’ (PGS). Indifference-zones are often used when comparing uncertain systems to reduce unnecessary effort by delineating a practical significant difference between the systems (Kim and Nelson, 2006), so that, for example, the selected system will have a performance within some δ^* of the best system, but need not actually be

[7]See (Fu, 2002) for a good overview of the Simulation Optimisation field. For general references to the area of R&S in Simulation Optimisation see, for example, (Bechhofer et al., 1995; Kim and Nelson, 2006). For an in-depth review of Bayesian methodology in Simulation Optimisation, see (Chick, 2006).

the best system.

Several approaches have been proposed for solving selection problems using sequential procedures: indifference-zone approaches (IZ) (Kim and Nelson, 2006); expected value of information procedures (VIP) (Chick and Inoue, 2001); and optimal computing budget allocation approaches (OCBA) (Chen, 1996). The methods differ principally in how the evidence for the correct selection is described, e.g. by frequentist or Bayesian statistics, and how simulation samples are allocated. The IZ procedures are frequentist based approaches that provide correct selection guarantees over repeated applications of a selection procedure; whereas Bayesian approaches, namely the VIP and OCBA, use posterior distributions to quantify the evidence for correct selection.

Branke et al. (2005, 2007) carried out an extensive empirical evaluation of the different approaches and a number of variants for the problem of identifying the best system. The comparison was carried out with respect to four criteria: *efficiency*, with respect to mean evidence for correct selection as a function of the mean number of samples; *controllability*, specifically the ease of setting the parameters of a procedure in order to achieve a target evidence level; *robustness*, the dependency of the effectiveness of the procedure on the underlying problem characteristics, i.e. the general applicability of the approach; *sensitivity*, the effect of the choice of parameter values on the mean number of samples required. With these considerations, the procedures were empirically evaluated on a number of problem sets, including structured and random problem instances. It was concluded that overall the Bayesian procedures are the most efficient and easiest to control (using new stopping criteria), in particular for non-stylised problem sets. Additionally, it is known that Bayesian approaches scale better for handling larger number of systems (Chen, 1996).

Based on the results from their study, Schmidt et al. (2006) and Schmidt (2007) proposed the tight integration of the OCBA with EAs for solving problems in stochastic/noisy environments. Due to the suitability of the framework for SCOPs, as described in the literature review in Section 3.4, the framework is applied here for solving the MSE problem.

We now detail the OCBA procedure after first providing additional notation and preliminaries. This is followed by a presentation of the integrated EA and OCBA framework.

5.4.1 Optimal Computing Budget Allocation

Originally proposed by Chen (1996), the OCBA approach is a sequential ranking and selection procedure based on Bayesian statistics. A number of variations of the OCBA have been proposed (Inoue et al., 1999; Chen et al., 2000; Branke et al., 2005, 2007; Chen et al., 2010) that differ in how they approximate the PCS and the heuristics they use for how additional samples might improve the PCS. The OCBA procedure uses posterior distributions to quantify the evidence for correct selection. Due to the complexity of computing the exact PCS, Chen showed that an efficient lower bound can easily be evaluated. The crux of the procedure is the rule for allocating additional simulations, or replications, at each stage, or loop, of the procedure. The procedure works by iteratively and greedily allocating additional samples to individuals where the largest improvement in the overall confidence in the selection is expected, where it is assumed the mean stays the same but that the standard error scales back accordingly.

5.4.2 Notation and Preliminaries

We now outline the basic notation and preliminaries for the OCBA method, including the underlying sampling assumptions. We follow the notation of Schmidt (2007).

Let Y_{ij} be a random variable whose realization y_{ij} is the output of the j-th evaluation of system i, for $i = 1,...,k$ and $j = 1,2,\ldots$, for k systems. Let w_i and ω_i^2 be the unknown mean and variance[8] of the performance of system i, and let $w_{[1]} \leq w_{[2]} \leq \cdots \leq w_{[k]}$ be the ordered means. In practice, due to uncertainty, the ordering $[\cdot]$ is unknown. The OCBA assumes that simulation output is independent and normally distributed, conditional on w_i and ω_i^2, for $i = 1,...,k$:

$$\{Y_{ij} : j = 1,2,\ldots\} \overset{i.i.d.}{\sim} \mathcal{N}(w_i, \omega_i^2)$$

In practice, the normality assumption is not always valid, however, it is often sufficient for the output to be approximately normal, in particular for large sample sizes.

Let n_i be the number of replications for system i executed so far. Let the sample mean be $\bar{y}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} y_{ij}$ and the sample variance $\hat{\omega}_i^2 = \frac{1}{n_i-1}\sum_{j=1}^{n_i}(y_{ij} - \bar{y}_i)^2$. Let $y_{(1)} \leq y_{(2)} \leq \cdots \leq y_{(k)}$ be the ordering of the sample means based on all simulations seen so far. Equality of sampled means is assumed to occur with probability 0. Given additional simulations, the quantities n_i, $\bar{y}_i$, $\hat{\omega}_i^2$ and ordering, (i), may change.

[8]Once again, to prevent confusion, σ^2 is not used to denote the population variance as is conventional.

From a Bayesian perspective, assuming uninformative priors, the means W_i of the systems are distributed according to the Student's t distribution[9]:

$$W_i \sim St(\bar{y}_i, \frac{n_i}{\hat{\omega}^2}, n_i - 1)$$

The upper case W_i is used to indicate the Bayesian perspective that w_i is a realisation of its corresponding random variable. The Student's t distribution is used when estimating the mean of a normally distributed population with unknown variance, and is more robust with small samples sizes. Using t distributions rather than Gaussians (using sample variance in place of the population variance) has been evaluated and no difference in efficiency was noted by Branke et al. (2007) and Chen et al. (2010).

In (Schmidt, 2007), several approximations were given for efficiently evaluating the difference between two Student's t variables. A method based on Welch's approximation (Welch, 1938) was found to outperform others, so that for individuals i and j, given evidence $\mathcal{E}$, we have

$$P(W_i > W_j | \mathcal{E}) \approx \Phi_{\nu_{ij}}(d_{ij} / \sqrt{s_{ij}^2})$$

where $d_{ij} = \bar{y}_i - \bar{y}_j$ is the observed difference, $s_{ij}^2 = \frac{\hat{\omega}_i^2}{n_i} + \frac{\hat{\omega}_j^2}{n_j}$, the sample variance of the estimated difference, and $\Phi_{\nu_{ij}}(\cdot)$ the cumulative distribution function of the Student's t distribution with degrees of freedom provided by Welch's approximation:

$$\nu_{ij} = \frac{\left((\hat{\omega}_i{}^2/n_i) + (\hat{\omega}_j{}^2/n_j)\right)^2}{(\hat{\omega}_i{}^2/n_i)^2/(n_i - 1) + (\hat{\omega}_j{}^2/n_j)^2/(n_j - 1)}$$

where $\min(n_i - 1, n_j - 1) \leq \nu_{ij} \leq (n_i + n_j - 2)$.

The classic selection problem is to identify the best system, individual $[k]$. Using the Slepian inequality[10], Welch's approximation, and given data $\mathcal{E}$ seen so far, the probability PCS_{Bayes} that the individual with the best observed mean, (k), is the individual with the best true mean, $[k]$, can be approximated:

$$\begin{aligned} PCS_{Bayes} &\doteq P\big(W_{(k)} \geq \max_{i \neq (k)} W_i | \mathcal{E}\big) \\ &\geq \prod_{i:(i) \neq (k)} P\big(W_{(k)} \geq W_{(i)} | \mathcal{E}\big) \\ &\approx \prod_{i:(i) \neq (k)} \Phi_{\nu_{(k)(i)}}(d_{(k)(i)} / \sqrt{s_{(k)(i)}^2}) \doteq PCS_{Slep} \end{aligned}$$

[9]The three-parameter, or shifted, Student's t distribution: $St(\mu, \kappa, \nu)$, forms a generalised location/scale family that introduces a location parameter μ and an inverse scale parameter (i.e. precision) $\kappa > 0$, as well as the degrees of freedom $\nu > 0$ (DeGroot, 1970). If $\nu > 2$, then the variance of three-parameter distribution is $\kappa^{-1} \cdot \nu/(\nu - 2)$.

[10]See, for example, (Kim and Nelson, 2006) for more details.

To include an indifference-zone, δ^*, so that the approximate probability that the difference between the selected system and the true best is no more than δ^*, we have:

$$PGS_{Slep,\delta^*} \doteq \prod_{i:(i)\neq(k)} \Phi_{\nu_{(k)(i)}}((d_{(k)(i)} + \delta^*)/\sqrt{s^2_{(k)(i)}})$$

5.4.3 OCBA Allocation Rule

The core of the OCBA procedure involves carrying out a 'thought experiment' in order to select a number of systems, q, that, given an additional number of simulations θ, are expected to best improve the overall confidence in the selection (expected to most improve the PGS_{Slep,δ^*} value, for example).

This idea is to hypothetically allocate a number of additional replications to a system, by simulating an increase in sample size, to reduce the variance in the estimate of the unknown mean through changing the precision from $\frac{n_i}{\hat{\omega}^2}$ to $\frac{(n_i+\theta)}{\hat{\omega}^2}$. The assumption is that the mean stays the same and that the standard error reduces accordingly.

The means of the systems $W_i, i = 1,\ldots,k$ with hypothetical allocations are then modelled:

$$St(\bar{y}_i, \frac{n_i}{\hat{\omega}^2}, n_i - 1) \text{ to } \bar{W}_i \sim St(\bar{y}_i, \frac{(n_i+\theta)}{\hat{\omega}^2}, n_i - 1 + \theta)$$

The effect of allocating an additional θ replications to a system i, but no samples to others, leads to an estimated approximate probability of good selection ($EAPGS_i$). The goal of the OCBA procedure, described next, is to iteratively and greedily allocate samples to those q systems that maximise

$$EAPGS_i - PGS_{Slep,\delta^*}$$

In other words, the aim is to allocate additional simulations to those q systems that individually promise the largest positive difference between the estimated approximate probability of good selection and the current PGS_{Slep,δ^*}.

5.4.4 OCBA for Selecting the Best System

For the selection-of-the-best problem, the $OCBA_{\delta^*}$ procedure is defined as follows (Schmidt, 2007). The procedure requires input parameters: the initial sample size, $\bar{n}_0$, the number of systems to allocate additional samples per stage, q, and the number of additional replications for each system, θ. Additionally, termination criteria are required, which may require further input parameters; they are described in the next section (5.4.5).

Procedure: $OCBA_{\delta^*}$

```
1. Run $\bar{n}_0 > 2$ independent replications for each system, $i = 1, \dots, k$.
2. Evaluate sample statistics, $\bar{y}_i$ and $\hat{\omega}_i$, and sample mean ordering $\bar{y}_{(1)} \leq \bar{y}_{(2)} \leq \dots \leq \bar{y}_{(k)}$.
3. WHILE Termination criteria not met DO
    (a) Compute $EAPGS_i$ for $i = 1, \dots, k$.
    (b) Allocate additional replications $n_i = n_i + \theta$ to the $q$ systems that maximise $EAPGS_i - PGS_{Slep,\delta^*}$, and none to the others.
    (c) Re-evaluate sample statistics, $\bar{y}_i$ and $\hat{\omega}_i$, and order statistics, so that $\bar{y}_{(1)} \leq \bar{y}_{(2)} \leq \dots \leq \bar{y}_{(k)}$.
4. Select system with best observed mean, $(k)$.
```

5.4.5 Stopping Rules for OCBA

Termination criteria must be defined as part of the OCBA procedure. The original OCBA (Chen, 1996) allocated additional replications given a total budget, S, while (Branke et al., 2005, 2007; Schmidt, 2007) suggested a new stopping rule based directly on the probability of good selection.

The rules are:

1. Sequential: repeat sampling if $\sum_{i=1}^{k} n_i < S$, for a specified total budget S.
2. Repeat while $PGS_{Slep,\delta^*} < 1 - \alpha^*$, for target probability $1 - \alpha^*$ and indifference-zone $\delta^* \geq 0$.

Branke et al. (2005, 2007) and Schmidt (2007) note that most previous work used rule 1, but rule 2 has been shown to improve the efficiency and controllability of OCBA approaches. In this work, where budget constraints are not fixed, rule 2 is adopted.

5.4.6 Integrating OCBA with EAs

For EAs in stochastic/noisy environments, Schmidt et al. (2006) and Schmidt (2007) proposed a framework for integrating statistical ranking procedures, in particular OCBA, with EAs. In this symbiosis, the EA is used to explore the solution space, and the

OCBA method is used to evaluate and rank individuals with the aim of reducing the effects of the noisy fitness evaluations on the functioning of the EA. Employing OCBA procedures due to the conclusions from their comparative study, they demonstrated that for typical EA configurations limiting the set of comparisons to those required can make ranking more efficient with respect to the total number of samples over, for example, a complete ranking of all individuals. Based on this, the OCBA approach was modified for ranking the set of comparisons required by the EA order information.

In the context of EAs, the ‘systems’ of the OCBA procedure are synonymous with the ‘individuals’ of the EA’s population. Their performance is analogous to the fitness of individuals, evaluated through a noisy fitness function. Therefore, running simulations or replications is analogous to resampling the noisy fitness function. The assumptions of the method in relation to the MSE are discussed in the following chapter, in Section 6.1.7.

5.4.7 EA Order Information

In the context of EAs, the correct functioning of the algorithm does not necessarily require a complete ranking of individuals. In fact, an EA requires that only a certain set of comparisons are correct, independently of whether all ranks are correct. In particular, Schmidt et al. (2006) and Schmidt (2007) note that there are specific steps of an EA that are effected by noise, namely, the *selection* (see description in Section 3.2.3) and *replacement* (3.2.7) steps (functions SelectMates and SelectReplacements respectively in the generic EA, Algorithm 3.2.1 described in Section 3.2). In deterministic environments, there will often be access to the complete ranking, and hence for pairs of individuals it is possible to determine which one is fitter with certainty. However, only sections of this information are actually used within an EA iteration. Therefore, to make the sampling procedure more efficient it would be useful to determine what information is necessary for the correct functioning of an EA. Schmidt (2007) provides the necessary sets of comparisons for the order information required by typical EA replacement strategies and selection operators, of which some are detailed here. Let the set of comparisons be denoted $\mathcal{C}$.

Recall that the population size of an EA is given by μ, and λ is the number of offspring created each generation. Then contained in the set $\mathcal{C}$ are those pairs of individuals that must be compared. Comparisons are specified as pairs $\langle i, j\rangle$ where $\forall \langle i, j\rangle \in \mathcal{C} : i, j \in \{1, \ldots, \mu+\lambda\}, i \neq j$. If the observed rank of individual i is higher than that of individual j, and i is actually better than j, then the comparison $\langle i, j \rangle$ is correct.

5.4.7.1 Replacement Strategies

Replacement strategies are used to decide which μ individuals out of $k = \mu + \lambda$ individuals from the current population and the offspring survive to the next iteration. Each individual, that is, each solution instance, can be added most once in the new population M, although note that this does not preclude other individuals with identical genotypes being added.

The individuals in the current population are indexed by $M' = \{1, \dots, \mu\}$, the offspring by $\Lambda = \{\mu + 1, \dots, k\}$, and the new population as the subset $M \subset \{1, \dots, k\}$, with $|M| = \mu$. The sample mean ordering $(\cdot)$ over all individuals is defined by $\bar{x}_{(1)} \leq \bar{x}_{(2)} \leq \cdots \leq \bar{x}_{(k)}$, over all offspring by $(\cdot)_\Lambda$, and over all individuals in the new population as $(\cdot)_M$. The orders $(\cdot)_\Lambda$ and $(\cdot)_M$ can be determined from the overall order $(\cdot)$ by finding the i-th individual having $(j) \in \Lambda$ or $(j) \in M$ for $j = 1, \dots, k$.

For generational replacement, the current population is completely replaced by the offspring, whereas generational replacement with elitism guarantees that the (observed) best from the current population survives and requires that only $\mu - 1$ offspring are generated each iteration to replace the current population minus the best. For the former approach, no comparisons are required, since the complete population is replaced, and therefore no comparisons are added to $\mathcal{C}$. The indices of the new population is the set of λ offspring:

$$M = \{\mu + 1, \dots, 2\mu\}$$

The latter approach adds comparisons between the best and all other individuals from the current population, so that the best individual is determined for the next iteration. The number of offspring generated is $\lambda - 1$ and the elite individual is $(\mu)_{M'}$.

$$\mathcal{C} \leftarrow \bigcup_{i=1}^{\mu-1} \{\langle (\mu)_M, (i)_M \rangle\}$$
$$M = \{(\mu)_{M'}, \mu + 1, \dots, 2\mu - 1\}$$

For (μ, λ) replacement with $\mu > \lambda \geq 1$, the best μ out of λ offspring survive to the next generation, so that we have comparisons and the new population given by

$$\begin{aligned} \mathcal{C} &\leftarrow \bigcup_{i=\lambda-\mu+1}^{\lambda} \{\langle (i)_\Lambda, (1)_\Lambda \rangle, \ldots, \langle (i)_\Lambda, (\lambda-\mu)_\Lambda \rangle\} \\ M &= \{(\lambda-\mu+1)_\Lambda, \ldots, (\lambda)_\Lambda\} \end{aligned}$$

For the $(\mu+\lambda)$ replacement strategy with $\mu, \lambda \geq 1$, the best μ out of a combined $\mu+\lambda$ individuals are selected. Note that the steady-state replacement operator typically uses $(\mu+1)$ replacement. Therefore, the new population and comparisons are determined by

$$\begin{aligned} \mathcal{C} &\leftarrow \bigcup_{i=\lambda+1}^{\mu+\lambda} \{\langle (i), (1) \rangle, \ldots, \langle (i), (\lambda) \rangle\} \\ M &= \{(\lambda+1), \ldots, (\lambda+\mu)\} \end{aligned}$$

5.4.7.2 Selection Operators

Selection operators are used to determine which of the current population are selected to be parents of the next set of offspring. Typical selection operators are mentioned in 3.2.3. Once again, we do not give an exhaustive list of the required comparisons for operators but list a few for demonstrative purposes. For a more comprehensive list, see Schmidt (2007).

For selection, π parents, denoted by the set P, are selected for mating from the new population, M. Note that the set of parents can contain duplicates.

The classic tournament selection operator, 2-tournament selection, chooses two individuals randomly from the population M and selects the better of the two as a parent. A few variants exist, including a more general t-tournament selection, that selects t individuals for each tournament, and stochastic tournament selection that selects two individuals and picks the better according to some probability, or else the worse of the two wins the tournament. For the general t-tournament selection, in terms of the comparisons required, once t individuals have been chosen, it is necessary that the individual with the highest fitness be chosen, and therefore it is necessary to add comparisons to guarantee this. Denote the chosen t-tournament participants of tournament j by $c_j^1, \ldots, c_j^t$ and by $c_j^{(i)}$ the ordered tournament participants, where $\bar{x}_{c_j^{(1)}} \leq \bar{x}_{c_j^{(2)}} \leq \cdots \leq \bar{x}_{c_j^{(t)}}$. Then the selected parents and the set of comparisons required are:

$$C \leftarrow C \bigcup_{j=1}^{\pi} \left\{ \left\langle c_j^{(t)}, c_j^{(1)} \right\rangle, \ldots, \left\langle c_j^{(t)}, c_j^{(t-1)} \right\rangle \right\}$$
$$P = \left\{ c_1^{(t)}, \ldots, c_\pi^{(t)} \right\}$$

Other potential selection operators include ranking operators, such as linear and exponential ranking. However, they require the complete ordering of individuals, which can be costly, and can, in fact, be replaced by variants of tournament selection that implement the same selection probability (see Schmidt (2007) for more information).

Finally, there is also the random selection operator commonly used with ESs that uniformly randomly selects individuals with replacement from the population to be parents. It does not require additional comparisons, and so adds no comparisons to $\mathcal{C}$.

5.4.7.3 Size of the Comparison Set

To appreciate the gain from using the set $\mathcal{C}$ here, consider that a complete ordering of k individuals requires $\binom{k}{2} = \frac{k(k-1)}{2} = O(k^2)$ comparisons. Assuming an $(\mu + (\lambda = \mu))$ EA[11], the number of comparisons for a full ranking of all individuals 2μ would be $\binom{2\mu}{2} = (2\mu \cdot (2\mu - 1))/2 = 2\mu^2 - \mu = O(\mu^2)$; while the number of comparisons in the set $\mathcal{C}$ is reduced to μ^2. Table 5.2 highlights the gain of using the set $\mathcal{C}$ over full ranking for various populations sizes using a $(\mu + \mu)$ EA. In practice, the set $\mathcal{C}$ requires approximately half the number of comparisons of a full ranking; however, it is still of the order $O(\mu^2)$. Therefore, in general the number of comparisons will not scale well with increasing μ, and thus it is advisable to keep the number of individuals for ranking low.

5.4.8 Using OCBA to Generate Order Information

Schmidt et al. (2006) and Schmidt (2007) showed that the OCBA can be further adapted for use with EAs and their respective ordering information. A new criterion, the ‘probability of good generation’ (PGG), was defined as the probability that all pairwise comparisons in $\mathcal{C}$ are correct. The following equation approximates a lower bound, using the same bounds and approximations as above, for the probability that for every pair in $\mathcal{C}$ the individual with the higher observed rank has a true mean that is

[11] $(\mu + \lambda)$ EA using elitist $(\mu + \lambda)$ replacement selection and random mating selection.

Table 5.2: Size of comparison set for $(\mu + \mu)$ EA

μ	Full Ranking	$\|\mathcal{C}\| = \mu^2$
10	$\binom{10+10}{2} = 190$	$10^2 = 100$
20	$\binom{20+20}{2} = 780$	$20^2 = 400$
50	$\binom{50+50}{2} = 4950$	$50^2 = 2500$
100	$\binom{100+100}{2} = 19900$	$100^2 = 10000$
250	$\binom{250+250}{2} = 124750$	$250^2 = 62500$
500	$\binom{500+500}{2} = 499500$	$500^2 = 250000$

higher the true mean of the other individual.

$$PGG_{Bayes} \doteq P\Big(\bigwedge_{\langle i,j\rangle \in \mathcal{C}} W_i + \delta^* > W_j | \mathcal{E}\Big) \tag{5.5}$$

$$\geq \prod_{\langle i,j\rangle \in \mathcal{C}} P\big(W_i + \delta^* > W_j | \mathcal{E}\big) \tag{5.6}$$

$$\approx \prod_{\langle i,j\rangle \in \mathcal{C}} \Phi_{\nu_{ij}}\big((\delta^* + d_{ij})/\sqrt{s^2_{ij}}\big) \doteq PGG_{\delta^*}(\mathcal{C}) \tag{5.7}$$

The OCBA procedure based on $PGG_{\delta^*}(\mathcal{C})$ allocation is called $\mathcal{OCBA}^{EA}_{\delta^*}$, and is defined next.

5.4.9 The $\mathcal{OCBA}^{EA}_{\delta^*}$ Procedure

Based on the new criterion, the following procedure was defined in Schmidt (2007) for the integration of the OCBA with EAs. The procedure has a number of input parameters: the initial sample size, $\bar{n}_0$; the number of individuals to allocate additional samples each stage, q; the number of additional samples per selected individual, θ; the indifference-zone, δ^*; and the target probability α^*. The parameters are discussed in detail in Section 5.4.12.

Procedure $\mathcal{OCBA}^{EA}_{\delta^*}(\bar{n}_0, q, \theta, \delta^*, \alpha^*)$:

1. Sample each unevaluated individual's fitness $\bar{n}_0$ times.

2. Determine sample statistics and rank individuals based on sampled fitness values.

3. Initialise $\mathcal{C}$: $\mathcal{C} \leftarrow \emptyset$, and add comparisons based on EA operators.

4. WHILE $PGG_{\delta^*}(\mathcal{C}) < 1-\alpha^*$ DO

   (a) Allocate samples θ to individuals q, according to $OCBA^{EA}_{\delta^*}$ allocation rule.

   (b) Update sample statistics and rank individuals based on sampled fitness values.

   (c) If ranks have changed from the previous iteration, reinitialise $\mathcal{C}$ (as step 3).

The $\mathcal{OCBA}^{EA}_{\delta^*}$ is called every iteration of the EA after the offspring have been generated and before replacement. After, the EA continues using the ordering based on the estimated fitness values to determine which individuals survive to the next generation, and so on.

5.4.10 An Optimisation of and a Heuristic for the OCBA

Here we contribute an extension to the $\mathcal{OCBA}^{EA}_{\delta^*}$ procedure with an optimisation and a heuristic for reducing the computational requirements of each stage of the procedure for a single selection problem.

In the context of an EA, a selection problem is carried out each generation of the algorithm, and therefore a potentially large number of selection problems will be executed. Furthermore, since, for an EA, the evaluation step is often the most computationally demanding step, overall it may be fruitful to optimise its computational requirements. Note that it is often assumed, in particular in the SO contexts for which the OCBA was originally designed, that sampling is much more computationally demanding than the OCBA procedure overhead (e.g. computing the $PGG_{Slep,\delta^*}(\mathcal{C})$ value); however, over many applications of the procedure and as the number of individuals increases, there may still be computational effort to save.

While these extensions are applied to the $\mathcal{OCBA}^{EA}_{\delta^*}$ procedure, in principle they could be adapted for the more general OCBA approach.

5.4.10.1 Faster Evaluation of Thought Experiment

The allocation rule of $\mathcal{OCBA}^{EA}_{\delta^*}$ requires the evaluation of the estimated approximate probability of good generation, $EAPGG_i$, for each individual, $i = 1,\ldots,k$, at each stage of the procedure. Depending on the EA configuration, for increasing number of individuals, the number of comparisons grows at different rates; however, as an example, for the $(\mu+\mu)$ EA described above, as with complete ranking, the number of comparisons will grow at a quadratic rate in the number of individuals, $|\mathcal{C}| = O(\mu^2)$. For this example, the basic procedure for evaluating the value $EAPGG_i$ for each individual, $i = 1,\ldots,2\mu$, at each stage of the procedure requires a total of $2\mu \cdot \mu^2 = O(\mu^3)$ comparison evaluations: for each of the 2μ individuals, the $EAPGG_i$ value must be evaluated for each individual over all μ^2 comparisons. We now detail a method to reduce the computation of $EAPGG_i$ to $O(\mu)$ for each individual $i = 1,\ldots,2\mu$, and $O(\mu^2)$ for all individuals, at the expense of $O(\mu^2)$ memory space. The main gain of this optimisation is reducing the number of comparison evaluations, and hence the required computation, across OCBA stages.

Define the subset of $\mathcal{C}$ that contains all comparisons involving individual $i' \in \{1,\ldots,k\}$, as $\mathcal{C}_{i'} = \{\langle i,j\rangle \in \mathcal{C} | i = i' \vee j = i'\}$. Since $\mathcal{C} = \mathcal{C}_{i'} \cup \mathcal{C}\backslash\mathcal{C}_{i'}$ and $\mathcal{C}_{i'} \cap \mathcal{C}\backslash\mathcal{C}_{i'} = \emptyset$ for $i' = 1,\ldots,k$, we have that

$$PGG_{\delta^*}(\mathcal{C}) = PGG_{\delta^*}(\mathcal{C}_{i'}) \cdot PGG_{\delta^*}(\mathcal{C}\backslash\mathcal{C}_{i'}) \tag{5.8}$$

In order to find promising individuals we evaluate what effect allocation of additional samples, θ, would have to each individual $i' = 1,\ldots,k$ independently of others. We denote the estimated approximate probability of good generation for each individual, $i' \in \{1,\ldots,k\}$, over the set $\mathcal{C}$ as $EAPGG_{i'}(\mathcal{C})$, and over the subset of $\mathcal{C}$, $\mathcal{C}_{i'}$, as $EAPGG_{i'}(\mathcal{C}_{i'})$. From Equation 5.8 and since allocating additional samples to an individual i' can only effect the comparisons in $\mathcal{C}_{i'}$, it follows that

$$EAPGG_{i'}(\mathcal{C}) \doteq EAPGG_{i'}(\mathcal{C}_{i'}) \cdot PGG_{\delta^*}(\mathcal{C}\backslash\mathcal{C}_{i'}) \tag{5.9}$$

Therefore, due to the fact that we know $PGG_{\delta^*}(\mathcal{C})$ and $PGG_{\delta^*}(\mathcal{C}_{i'})$ from the previous OCBA sequential stage, it is possible to exploit this fact to simplify the calculations required for each stage's allocation of replications. This is done for each individual $i' = 1,\ldots,k$, by instead evaluating

$$EAPGG_{i'}^{OPT}(\mathcal{C}) \doteq PGG_{\delta^*}(\mathcal{C}) \cdot \frac{EAPGG_{i'}(\mathcal{C}_{i'})}{PGG_{\delta^*}(\mathcal{C}_{i'})} \tag{5.10}$$

We can therefore save computational effort by using available values from the previous OCBA stage, and only evaluate $EAPGG_{i'}(\mathcal{C}_{i'})$ for each individual, $i' = 1,\ldots,k$.

Proposition 1. *For a comparison set $\mathcal{C}$ and for $i' = 1,\ldots,k$,*

$$EAPGG_{i'}(\mathcal{C}) = EAPGG_{i'}^{OPT}(\mathcal{C})$$

Proof.

$$\begin{aligned}
EAPGG_{i'}(\mathcal{C}) &= EAPGG_{i'}(\mathcal{C}_{i'}) \cdot PGG_{\delta^*}(\mathcal{C} \backslash \mathcal{C}_{i'}) && \text{(Equation 5.9)} \\
&= \Big(EAPGG_{i'}(\mathcal{C}_{i'}) \cdot PGG_{\delta^*}(\mathcal{C} \backslash \mathcal{C}_{i'})\Big) \cdot \frac{PGG_{\delta^*}(\mathcal{C}_{i'})}{PGG_{\delta^*}(\mathcal{C}_{i'})} \\
&= \frac{EAPGG_{i'}(\mathcal{C}_{i'}) \cdot PGG_{\delta^*}(\mathcal{C} \backslash \mathcal{C}_{i'}) \cdot PGG_{\delta^*}(\mathcal{C}_{i'})}{PGG_{\delta^*}(\mathcal{C}_{i'})} \\
&= \frac{EAPGG_{i'}(\mathcal{C}_{i'}) \cdot PGG_{\delta^*}(\mathcal{C})}{PGG_{\delta^*}(\mathcal{C}_{i'})} && \text{(by Equation 5.8)} \\
&= PGG_{\delta^*}(\mathcal{C}) \cdot \frac{EAPGG_{i'}(\mathcal{C}_{i'})}{PGG_{\delta^*}(\mathcal{C}_{i'})} && \text{(Equation 5.10)} \\
&= EAPGG_{i'}^{OPT}(\mathcal{C})
\end{aligned}$$

□

Now to find the individuals that are expected to make the best improvement, the allocation rule uses Equation 5.10, such that for $i' = 1, \ldots, k$, the goal is to maximise:

$$EAPGG_{i'}^{OPT}(\mathcal{C}) - PGG_{\delta^*}(\mathcal{C})$$

This optimisation reduces the number of comparison evaluations for each stage of the OCBA procedure when evaluating the estimated approximate probability of good generation for each individual, and hence reduces the overall computational requirements of the procedure. This improvement can have a significant impact for selection problems with many stages and over a series of selection problems.

The additional memory space required by this extension is $O(k^2)$. The sets $\mathcal{C}_{i'}$ for $i' = 1, \ldots, k$ are determined once per stage when (re-)initialising the set $\mathcal{C}$ (step 3 and 4c of the $\mathcal{OCBA}_{\delta^*}^{EA}$ procedure above), which requires $O(k^2)$ space. Also, the $PGG_{\delta^*}(\mathcal{C}_{i'})$ values, $i' = 1, \ldots, k$, are stored when evaluating the full $PGG_{\delta^*}(\mathcal{C})$ value (required by step 4 of the procedure), requiring $O(k)$ space. So, overall, the memory requirement is $O(k^2)$.

5.4.10.2 Heuristic for Faster Selection of Best q Individuals

At each stage of the OCBA procedure q individuals are selected for allocation of additional samples, and a heuristic is proposed here for reducing the computation further in the selection of these individuals. The idea is to maintain an ordered list of the current best q $\frac{EAPGG_{i'}(\mathcal{C}_{i'})}{PGG_{\delta^*}(\mathcal{C}_{i'})}$ values, $(p_1, p_2, \ldots, p_q)$, where $p_1 \geq p_2 \geq \cdots \geq p_q$. Then, when evaluating the $EAPGG_{i'}(\mathcal{C}_{i'})$ value for an individual i', its comparisons are sorted into ascending order by their previous stage's probability of being correct, and after each comparison, after the hypothetical value has been evaluated, the current $\frac{EAPGG_{i'}(C_{i'})}{PGG_{\delta^*}(C_{i'})}$ is evaluated against the current least-best value, p_q. Since the value of $EAPGG_{i'}(C_{i'})$ is nonincreasing over the set $C_{i'}$, if the current $\frac{EAPGG_{i'}(C_{i'})}{PGG_{\delta^*}(C_{i'})}$ value is less than p_q then the process can stop and move to the next individual, $i'+1$, and so on. This reduces the number of comparisons evaluated for individuals that will not be amongst the selected q individuals, as soon as there is evidence that indicates they will not be chosen.

5.4.11 Informative Priors for Series of Selection Problems

To the knowledge of the author, until now, in the context of EAs, the OCBA has only been evaluated for solving a single selection problem. However, for an EA, over a full run, a selection problem is solved at each generation of the algorithm. The input to each selection problem may include individuals from previous iterations that have already been evaluated, and hence have previously been allocated a number of samples. For individuals with no samples, uninformative priors are assumed; however, for individuals with existing samples informative priors are used instead.

Before any evidence is provided, given the assumptions of unknown mean and precision (reciprocal of the variance), uninformative priors are assumed, so that the posterior marginal distribution of the unknown mean, W_i, where the data $\mathcal{E}_i = \{y_{i1}, \ldots, y_{i\bar{n}_0}\}$ is assumed (approximately) normally distributed, is modelled by:

$$W_i \sim St(W_{i\bar{n}_0}, \bar{n}_0\lambda_{i\bar{n}_0}, \bar{n}_0)$$

where $W_{i\bar{n}_0} = \bar{y}_{i\bar{n}_0}$ is the sample mean of the data, $\lambda_{i\bar{n}_0} = \hat{\omega}_{i\bar{n}_0}^{-2}$ is the sample precision, and $\bar{n}_0$ is the number of initial samples. As previously mentioned, using a Student's t distribution is consistent with the assumption of unknown variance and is more robust with the use of small samples sizes, e.g. $\bar{n}_0 < 30$.

Based on Inoue and Chick (1998) and Schmidt (2007), but not previously applied to the $\mathcal{OCBA}_{\delta^*}^{EA}$, after each subsequent stage of sampling, priors are modelled using normal-gamma conjugate prior distributions. For the priors, the marginal distribution of $\lambda_{i\bar{n}_i}$ is modelled:

$$P(\lambda_{i\bar{n}_i}) \sim \mathcal{G}(\alpha_i, \beta_i)$$

where $\alpha_i = \frac{\bar{n}_0}{2}$ and $\beta_i = \frac{1}{2}\sum_{j=1}^{\bar{n}_0}(y_{ij} - \bar{y}_{i\bar{n}_0})^2$, and the conditional distribution of W_i given $\lambda_{i\bar{n}_i}$ is:

$$P(W_i|\lambda_{i\bar{n}_i}) \sim \mathcal{N}(\bar{y}_{i\bar{n}_0}, \bar{n}_0\lambda_{i0})$$

From (DeGroot, 1970), the posterior marginal distribution of W_i given the above prior distributions and new evidence $\mathcal{E}_i = \{y_{i1}, \ldots, y_{i\bar{n}_i}\}$, for $\bar{n}_i$ additional samples, is then the three-parameter Student's t distribution:

$$P(W_i|\mathcal{E}_i) \sim St(W_{i\bar{n}_i}, \frac{(\bar{n}_{i0}+\bar{n}_i)\alpha_{i\bar{n}_i}}{\beta_{i\bar{n}_i}}, 2\alpha_{i\bar{n}_i})$$

where

$$\begin{aligned}
W_{i\bar{n}_i} &= \frac{\bar{n}_0 W_{i0} + \bar{n}_i \bar{y}_{i\bar{n}_i}}{\bar{n}_0 + \bar{n}_i} \\
\alpha_{i\bar{n}_i} &= \alpha_i + \frac{\bar{n}_i}{2} \\
\beta_{i\bar{n}_i} &= \beta_i + \frac{\sum_{j=1}^{\bar{n}_i}(y_{ij} - \bar{y}_{i\bar{n}_i})^2}{2} + \frac{\bar{n}_{i0}\bar{n}_i(W_{i0} - \bar{y}_{i\bar{n}_i})^2}{2(\bar{n}_{i0}+\bar{n}_i)}
\end{aligned}$$

Using informative priors effectively allows the OCBA to re-use samples from previous iterations, which becomes particularly helpful for individuals that survive over many EA generations. The allocation rule for the $\mathcal{OCBA}^{EA}_{\delta^*}$ is now modified to use substituted values: $\hat{\omega}_i^2 = \frac{\beta_{i\bar{n}_i}}{\alpha_{i\bar{n}_i}}$, $n = \bar{n}_{i0} + \bar{n}_i$ and $\nu = 2\alpha_{i\bar{n}_i}$; and all additional samples are allocated using informative priors as well.

5.4.12 $\mathcal{OCBA}^{EA}_{\delta^*}$ Parameters

The $\mathcal{OCBA}^{EA}_{\delta^*}$ procedure has a number of parameters, which are listed in Table 5.3. The literature provides some general guidelines on effective ranges and values for parameters. However, note that on the whole these guidelines have been laid down for the use of the OCBA in SO. In these contexts, where simulations dominate in computational effort, and for the execution of a single selection problem, not necessarily a series of selection problems as is the case for an integrated EA approach. We now discuss each parameter individually.

Schmidt (2007) found that Bayesian procedures are quite sensitive to the size of $\bar{n}_0$. It is well understood that the initial sample size should not be too small, as this may lead to poor estimates of the mean and variance, which would result in poor computing budget allocation (i.e. allocation of subsequent samples). However, the value should not be too large either, which would result in a waste of computation spent on non-promising individuals and, as this value is increased, would tend to increasingly obviate the OCBA altogether. Bechhofer et al. (1995) recommends a value between 10 and 20.

For the number of individuals selected at each stage, q, if we set $q = 1$ and $\theta = 1$ then the procedure is known as fully sequential — one individual is chosen each stage of the OCBA and allocated a single additional sample; this is common in SO where simulation runs may be very costly, however, this is not necessarily the case here. At the other extreme, if $q = k$ then this equivalent to equal allocation of additional samples, which is equivalent to not using the OCBA method, since all individuals are allocated the same number of samples. Chen et al. (1997) found that OCBA is not too sensitive to the size of q as long as θ is small. As a heuristic, Chen et al. (1999) recommend $q \in [k/20, k/10]$.

It is generally understood that θ should be neither too large nor too small: it is desirable to avoid, on the one hand, devoting unnecessary computational effort to reach an unnecessarily high confidence level and, on the other, requiring too many stages of the procedure. Also, importantly, the value should not be so large as to diminish the usefulness of the heuristic allocation rule. If a relatively large value of θ seems appropriate, it may be better to increase $\bar{n}_0$ instead.

Schmidt (2007) also showed that Bayesian procedures are quite sensitive to the size of the indifference-zone, δ^*. The indifference-zone is used to set the practically significant difference between solutions; that is, it may suffice to find a solution that is not the optimal solution, but whose quality is within some (practically insignificant) margin of the best. It is important not to set this value too small, since computation may be wasted on non-promising solutions; on the other hand, too large an indifference-zone may result in significantly sub-optimal solutions. Since judging significance is problem dependent, so too is the value of this parameter. Note that this work uses the same value for the indifference-zone used for the stopping rule and for evaluating the PGG_{δ^*} value during the selection.

The choice of α^* value sets the target confidence level: $1-\alpha^*$. For single selection problems (in SO), typically, a value is selected from $\{0.1, 0.05, 0.01\}$, in correspondence with conventional statistical confidence levels. Setting a low α^* value, and hence a high $1-\alpha^*$ value, is equivalent to requiring a high level of confidence in the selection; naturally, this increases the required computation. Again, setting this value too high may waste computation on non-promising solutions.

For integration with EAs, Schmidt (2007) highlights the importance, in particular, of the values of α^* and δ^*. The difficulty in setting these parameters for $\mathcal{OCBA}^{EA}_{\delta^*}$ is compounded by the use of the this procedure (and its parameter values) across all iterations of the EA. For example, in early runs of an EA, where solutions are typically of low quality, a high confidence in the ordering, e.g. $\alpha^* = 0.01$, may not be necessary; whereas as the iterations progress, and the algorithm begins to focus on better areas of the search space, the significance of the correct ordering increases. A similar situation arises for δ^*; if the value is too small then computational effort may be wasted in discriminating between non-promising solutions in early runs. The optimal set of parameters would allow the selection procedure to both minimise the number of allocated samples and procedure stages required to meet the confidence level appropriate for the phase of the EA's evolution, while still allowing the EA to find good solutions or even the optimal solution in the minimum number of generations. However, this may be difficult in general using a predefined, static set of parameter values.

The parameter values used for empirical evaluation of the approach are given in Chapter 7.

Table 5.3: $\mathcal{OCBA}^{EA}_{\delta^*}$ Parameters

Parameter	Range	Description
$\bar{n}_0$	$\mathbb{N}_{>2}$	initial sample size
q	$[1,k]$	number of individuals to select at each stage
θ	$\mathbb{N}$	number of additional samples for each of q individuals
δ^*	$\mathbb{R}$	indifference-zone
α^*	$[0,1)$	target confidence level: $1-\alpha^*$

5.5 Summary

This chapter presents efficient methods for generating, evaluating solution quality and comparing candidate solutions to the MSE. Given the computational complexity of the MSE problem, a metaheuristic approach is proposed herein, and therefore methods are required to generate candidate solutions to the MSE. However, generating solutions in a straightforward manner from the underlying topological graph may bias solutions towards certain regions of the solutions space. To counter this, an approach was presented that samples solutions with approximately equal probability from the solution space.

In general, evaluating solutions for the MSE is a computationally hard problem, involving calculations over potentially high-dimensional spaces. Therefore, as proposed in the literature, a simulation approximation method was presented for evaluating the solution quality of solutions based on well-known Monte Carlo sampling methods. Using these methods, we can show asymptotic convergence of the estimated quality to the true solution quality. Furthermore, two sampling frames were discussed and a procedure for sampling from a space of lower dimensionality than an approach from the literature was proposed, improving the efficiency of sampling in practice.

In the context of optimisation, as well as evaluating the quality of individual candidate solutions, it is also necessary to discriminate between them. However, the aforementioned approximate evaluation approach introduces potential selection error, so that with finite resources it is not possible to differentiate between candidate solutions with certainty, with respect to solution quality. In order to effectively compare candidate solutions with uncertain evaluations, it was proposed to apply a framework integrating a state-of-the-art statistical ranking and selection procedure, the Optimal Computing Budget Allocation (OCBA) approach, with Evolutionary Algorithms. The framework was explicated, introducing the OCBA approach in general, and then the integrated framework. Additionally, an optimisation for and a heuristic extension to the OCBA were proposed for improving the computational performance of the procedure. Furthermore, to increase the efficiency of using the OCBA for a series of selection problems, the procedure was modified for use with informative Bayesian priors.

Overall, this chapter has proposed efficient methods for generating, evaluating and ranking candidate solutions for the MSE that are required by the solution method proposed herein. The following chapter details the main approach.

Chapter 6

Solution Approaches

This chapter lays out the details of the main solution approach proposed for solving the Maximal Safest Escape Problem in STV networks. Given the problem's complexity, it is unlikely that a general, efficient solution approach will be presented. In this work, a metaheuristic approach based on a standard $(\mu + \lambda)$ Evolutionary Algorithm (EA) integrated with the $\mathcal{OCBA}^{EA}_{\delta^*}$ procedure is proposed. For comparison with the main EA approach, an EA variant and a baseline random search approach are also presented.

Firsty, the main EA approach is detailed, including the various components of the algorithm; then, secondly, the EA variant is described; and, finally, the simple random search algorithm is presented.

6.1 Noisy Evolutionary Algorithm

The main approach proposed for solving the MSE is a based on an $(\mu + \lambda)$ Evolutionary Algorithm (Bäck et al., 1997). The reasons for proposing a metaheuristic approach are two-fold. Firstly, given the complexity of the problem (as discussed in Section 4.3.3), it is unlikely that efficient methods can be developed for solving the problem optimally, or even approximating the solution within a constant factor of the optimal solution.

Secondly, metaheuristic approaches, in particular EAs, have been applied to a number of NP-hard combinatorial optimisation problems, both deterministic and stochastic, with some success (see sections on COPs (3.3) and SCOPs (3.5) for more details). While EAs are not guaranteed to find optimal solutions, they can often find good quality solutions for hard problems using significantly less effort than exact algorithms (Gendreau and Potvin, 2010). For evaluation here, the main approach is compared against a simple EA, a baseline approach and upper bounds on optimal solutions in order to provide a reference for the performance of the metaheuristic.

The proposed algorithm described by Algorithm 6.1.1 is based on the Generic EA described in 3.2 extended with integration of the $\mathcal{OCBA}_{\delta^*}^{EA}$ procedure (as defined in 5.4.9). Since the EA utilises a noisy fitness function, it is henceforth called a Noisy EA (NEA).

Algorithm 6.1.1: $(\mu+\lambda)$ NOISY EVOLUTIONARY ALGORITHM(Π)

```
$i \leftarrow 0$
$\mathcal{P}_i \leftarrow$ InitialisePopulation($\mu$)
$\mathcal{OCBA}_{\delta^*}^{EA}(\bar{n}_0, q, \theta, \alpha^*, \mathcal{P}_i)$
**while** $i < \kappa$
  **do** {
    $i \leftarrow i+1$
    $\bar{\mathcal{P}} \leftarrow$ SelectMates($\mathcal{P}_{i-1}, \lambda$)
    $O \leftarrow$ Crossover($\bar{\mathcal{P}}, p_c$)
    Mutate($O, p_m$)
    $\mathcal{OCBA}_{\delta^*}^{EA}(\bar{n}_0, q, \theta, \alpha^*, \mathcal{P}_{i-1} \cup O)$
    $\mathcal{P}_i \leftarrow$ SelectReplacements($\mathcal{P}_{i-1} \cup O, \mu$)
  }
```

The set Π is the collection of input parameters to the algorithm. For the $(\mu+\lambda)$ NEA, the set is $\Pi = \{\mu, \lambda, \kappa, p_c, p_m, \bar{n}_0, q, \theta, \delta^*, \alpha^*\}$. The EA parameters, discussed in Section 3.2, are the subset $\{\mu, \lambda, \kappa, p_c, p_m\}$, and for the $\mathcal{OCBA}_{\delta^*}^{EA}$ procedure the set $\{\bar{n}_0, q, \theta, \alpha^*, \delta^*\}$, discussed in Section 5.4.12.

As a reminder, an $(\mu+\lambda)$ EA uses a population of size μ and an offspring size λ. At iteration i, the current population is denoted by $\mathcal{P}_{i-1}$, the current offspring O, and the new population by $\mathcal{P}_i$. To simplify algorithm parameters, the number of offspring is set $\lambda = \mu$, so that the algorithm is a $(\mu+\mu)$ NEA.

The following sections describe the necessary particulars for the NEA: the candidate solution representation, the initialisation procedure, the neighbourhood function, the crossover operator, the mutation operator, the selection strategies: mating and replacement schemes, the fitness evaluation method, and, finally, the stopping criterion. A basic description of these components is given in Section 3.2.

6.1.1 Candidate Solution Representation

The solution space for the MSE's objective (Equation 4.1) is the set of all flow patterns Ψ. As defined in the preliminaries (Section 4.1.5), each solution, or flow pattern, $\psi \in \Psi$, is a multiset of pairs of a path and a departure time, and a multiplicity function determining the units of flow assigned to each pair. For the EA, it is not necessary for the chromosome representation to be equivalent to the solution representation of the optimisation problem. In practice, there are often a number of ways to represent these formal objects; for example, tuples of integers can be represented by both binary strings and integer-valued vectors. Nevertheless, if suitable, it is often simpler to represent candidate solutions using canonical EA representations so that standard genetic operators and constraint handling methods are available.

For the MSE problem in particular, there are various ways to represent candidate solutions, and several works, surveyed in the literature review Sections 3.3 and 3.5, have addressed EA representations for flow problems (Munakata and Hashier, 1993; Sadek et al., 1997; Varia and Dhingra, 2004; Opasanon, 2004; Opasanon and Miller-Hooks, 2010; Miller-Hooks and Sorrel, 2008).

We briefly discuss different representations used by related works and then detail the chosen representation.

6.1.1.1 Survey of Representations

In network flow optimisation in general (Ahuja et al., 1993), solutions for shortest path and flow problems are typically represented by a flow matrix, where each entry in the matrix represents the flow associated with each edge in the network, possibly over time; for example, the number of units departing an edge (i, j) at time t, $x_{ij}(t)$. This was the EA representation used by Munakata and Hashier (1993) for solving the classic static maximum flow problem[1], where each chromosome, or solution, is a single flow matrix. However, this representation is not efficient in general for EAs since it includes all edges in the network, and hence the flow matrix has size $O(v^2)$, or over time, $O(v^2 T)$, and requires special genetic operators to handle problem constraints.

A more common, and perhaps simpler approach, is a representation using ordered lists, or vectors. Sadek et al. (1997) used real-valued vectors where each entry in a vector represents the number of vehicles assigned to an edge during a time period. Constraints are handled though a penalty on the fitness function. In contrast, Varia and

[1]For details on the canonical maximum flow problem, see, for example, Ahuja et al. (1993).

Dhingra (2004) used a binary string representation that encoded each $s-l$ path in the network with a level of flow assigned to it. This representation is also inefficient in general since it encodes all paths in the network. Again a penalty function is used for penalising infeasible candidate solutions.

Works more closely related to the MSE problem, Opasanon (2004) and Opasanon and Miller-Hooks (2010) opted for variable-length vector representations, where each entry in the vector represents a pair, consisting of a path in the network and an associated flow. This approach closely matches the solution representation described for their respective optimisation problem, and is compact, neither representing all paths in the network nor allowing multiple duplicate path entries for common departure times. Still, genetic operators must be designed to handle feasibility issues. Miller-Hooks and Sorrel (2008) extended the representation so that each entry of the vector is itself a variable-length vector. Again special operators are designed to cope with feasibility.

A number of similar representations have been suggested for solving deterministic and stochastic shortest path problems, including permutation-based encodings, and fixed- and variable-length vectors of nodes or edges (Gen et al., 1997; Inagaki et al., 1999; Ahn and Ramakrishna, 2002; Ji, 2005; Davies and Lingras, 2003).

The literature indicates that a number of representations can be used for flow problems, and that often genetic operators are required to be designed to handle feasibility issues.

6.1.1.2 MSE Representation

The representation used here, which is similar to that of Opasanon (2004) and Opasanon and Miller-Hooks (2010), is equivalent to the abstract solution representation detailed in Section 4.1.5. Due to this, the same notation is used to represent MSE candidate solutions and EA candidate solutions. For the EA then, an individual flow pattern, or candidate solution, corresponds to one of the "chromosomes" that make up the algorithm's population, and each pair, consisting of a path in the network and a departure time, that is part of the flow pattern is represented using one "gene" for each occurrence (multiplicity)[2]. For example, for a multiset (R,m), the EA chromosome is represented by $n = |(R,m)| = \sum_{r\in R} m(r)$ genes, where $\{r_1, r_2, \ldots, r_k\} \subseteq R$ and $m(r_i) > 0$, $i = 1, \ldots, k \leq n$, so that, using an alternative multiset representation[3], the collection of

[2]Pairs with multiplicity of 0 are thus not represented.
[3]For more details on multiset representations, see, for example, Singh et al. (2007).

n genes is:

$$\{r_{1,1}, r_{1,2}, \ldots, r_{1,m(r_1)}, r_{2,1}, \ldots, r_{2,m(r_2)}, \ldots, r_{k,1}, \ldots, r_{k,m(r_k)}\}$$

One advantage of this approach is that it is relatively compact, representing only those portions of the network that are relevant to a particular solution, versus, for example, a flow matrix representation. However, as with the reviewed representations, it too requires specially designed operators, in particular for issues of feasibility. These operators are described in the following sections.

6.1.2 Initialisation

Each execution of an EA requires an initial population with which to seed the search process. In the absence of information about the search space, a common approach is to sample the search space uniformly, so that the initial candidate solutions are uniformly distributed across the search space. Hence, all candidates, and hence all genes, have a probability greater than 0 of getting sampled. When knowledge is available, one approach is to seed the population using, for example, constructive heuristics or greedy methods, in order to introduce feasible or reasonable quality candidates.

For the MSE, due to the complexity of evaluating exact solution quality, in general it is difficult to generate initial candidates of reasonable quality over all flow pattern states. However, in order to reduce the space from that of all flow patterns, candidate solutions are generated with a minimal consideration for feasibility. In particular, the initialisation procedure generates only candidates that observe flow conservation (the supply/demand structure) by generating enough paths and multiplicity to convey flow from source to sink to satisfy $b_s(t)$, for $t \in S$, in an unconstrained, static network, that is, without consideration for time and edge properties: capacities and travel times, because these properties introduce substantial complexity for evaluation.

The initial EA population $\mathcal{P}_0$ is generated from this constrained solution space; that is, a set of μ chromosomes is sampled from the constrained chromosome space. Even though it is difficult to generate solutions that are of high quality over all flow pattern states, the procedure proposed to solve Deterministic MSE problems defined in Chapter 4, in Section 4.4.3.2, is used to generate an initial population where each solution is at least feasible on a randomly generated network state. The procedure is used to solve μ Deterministic MSE problems, resulting in, in the best-case, μ candidate solutions for the MSE problem that form the initial population[4]. As is shown in the following

[4]The procedure cannot guarantee to find a feasible flow for a particular sampled network state;

chapter, in Section 7.4, the seeding of the initial populations plays a significant role in the successful application of the NEAs to the MSE.

If the number of candidate solutions generated is less than the required population size, μ, then to guarantee that a total of μ candidates are generated, the remaining candidate solutions are approximately uniformly randomly generated from the constrained space. The procedure for this is as follows. Each chromosome consists of a number of genes, where each gene is a pair consisting of a path and a departure time; thus, in order to generate equally likely candidate solutions from the constrained space, a sequence of B paths is randomly generated with $b_s(t)$ paths assigned to each departure time with available supply, $t \in S$. Due to the multiset representation, in the case that identical paths are assigned to the same t, they are replaced by a single path with multiplicity equal to the total flow assigned to the individual, identical paths. Paths are found using the weighted depth-first path-finding procedure described in Section 5.1.3.

Finally, the algorithm's genetic operators: crossover (6.1.4) and mutation 6.1.5, were designed to maintain this level of feasibility.

6.1.3 The Neighbourhood

An important decision in the development of metaheuristic algorithms is the definition of the neighbourhood function, $N : \Psi \to 2^{\Psi}$, that assigns to each candidate solution, $\psi \in \Psi$, a set of candidate solutions $N(\psi) \subset \Psi$ (Gendreau and Potvin, 2010). A candidate solution ψ' in the neighbourhood of ψ, $\psi' \in N(\psi)$, is called a neighbour of ψ. The neighbourhood function defines a structure over the solution space, and a connectivity or a metric distance between candidate solutions.

Associated with the neighbourhood function is a move operator that performs a small change to a candidate, say ψ, to transform, or "move", it to one of its neighbours, $\psi' \in N(\psi)$. The move operator is utilised by metaheuristics to search through the solution space by moving candidates to candidate solutions in their neighbourhood. Here the neighbourhood structure and move operator, which are formally defined in the following sections, are used by the EA crossover operator (6.1.4) for generating offspring, and the EA mutation operator (6.1.5) for perturbing offspring — both operations effectively corresponding to moves in the solution space.

Applying the move operator a finite number of times, $k \in \mathbb{N}$, to any solution in the search space should be able to move it to any other solution in the space. In other

however, experience indicates that should a state pass the heuristic checks for infeasibility, the greedy method finds solutions a majority of the time.

words, all candidate solutions must be in each other's k-neighbourhoods, so that a metric is defined over the neighbourhood structure whereby for two candidate solutions, say ψ_1 and ψ_2, the distance between them is the minimum number of applications of the move operator required to "move" ψ_1 to ψ_2. If it were not the case that all candidate solutions are reachable, then the problem may not be solvable, that is, potential solutions or even regions of the search space may exist that the algorithm cannot reach.

The neighbourhood function for the multiset representation is formally defined in the following section.

6.1.3.1 Multiset Preliminaries

Here we introduce a number of preliminaries required for the definition of the neighbourhood function that underlies the genetic operators of the EA.

Formally, a multiset is a pair (A,m), where A is some set and m is a function $m : A \rightarrow \mathbb{N}_0$, which defines the number of occurrences of each member of A in the multiset. The set A is called the set of underlying elements. The cardinality of (A,m) is $|(A,m)| = \sum_{a\in A} m(a)$. A submultiset (B,n) of a multiset (A,m) is a subset $B \subseteq A$ and a function $n : B \rightarrow \mathbb{N}_0$ such that $\forall b \in B : n(b) \leq m(b)$. The union of two multisets is defined $(A \cup B, f)$, where $\forall x \in (A \cup B), f(x) = \max\{m(x), n(x)\}$, and the intersection is $(A \cap B, g)$, where $\forall x \in (A \cap B), g(x) = \min\{m(x), n(x)\}$. Two multisets are equal if the sets of underlying elements are equal, $A = B$, and $\forall a \in A, m(a) = n(a)$, and equivalently for the elements in B.

The power set[5] of (A,m), denoted $\mathcal{P}\big((A,m)\big)$, is the set of all submultisets of (A,m), including the empty set and (A,m) itself. The set $\mathcal{P}\big((A,m)\big)$ has cardinality $\leq 2^{|(A,m)|}$, where equality occurs only if (A,m) is a set.

The symmetric difference between two sets, Y and Z, denoted $Y\Delta Z$, is the set of elements that are in Y or Z but not in their intersection: $Y\Delta Z = (Y \cup Z)\backslash(Y \cap Z)$. For multisets, it is extended to include multiplicities, and defined $(A\Delta B, f)$ with $f(x) = \max\{m(x), n(x)\} - \min\{m(x), n(x)\} = |m(x) - n(x)|$, $\forall x \in A\Delta B$. The symmetric distance between multisets is then

$$d\big((A,m),(B,n)\big) = |(A\Delta B, f)| = \sum_{x\in A\Delta B} f(x) = |m(x) - n(x)|$$

Based on these definitions, we now define the neighbourhood function.

[5]For more details on power sets and power multisets, see, for example, Singh et al. (2007).

6.1.3.2 Neighbourhood function

The symmetric distance between candidate solutions is taken as a metric on the search space, and provides the formal basis for the neighbourhood function, N. The neighbourhood of a candidate solution ψ, $N(\psi)$, is defined as the set of candidate solutions that have a single unit of symmetric distance from ψ:

$$N(\psi) = \{\psi' \in \Psi : d(\psi, \psi') = 1\}$$

Therefore, a single application of the move operator based on the neighbourhood function, N, will “move” or transform a candidate to a another candidate in its neighbourhood that comprises of candidate solutions that have a symmetric distance of 1.

Based on a geometric interpretation of genetic operators, Moraglio and Poli (2004) attempted to unify the theoretical foundations of EA operators, whereby crossover and mutation are based on the same neighbourhood structure. Here that structure is defined using symmetric difference and distance. The definition of the crossover operator is given next, and the mutation operator follows.

6.1.4 Crossover

In each generation, the crossover operator, defined next, is applied to pairs of parents selected by the mating selection operator (described in Section 6.1.6.1) to generate pairs of offspring.

Several existing works have addressed the question of genetic operators for multiset representations. Early works (Radcliffe, 1991c, 1992; Radcliffe and George, 1993) applied Radcliffe’s Forma Analysis (Radcliffe, 1991a,b) to recombination for set representations. Later, Moraglio and Poli (2004) laid out a geometric framework for interpreting crossover and mutation in an attempt to unify the theoretical foundations of EA genetic operators. They argued that their geometric interpretation simplifies the relationship between EA search operators and the fitness landscape, and leads to a general and intuitive interpretation of the operators that are likely to perform well. In (Moraglio and Poli, 2004), they showed that a number of existing traditional genetic operators for binary strings fit within this framework. Later, in (Moraglio and Poli, 2006), they proposed a uniform geometric crossover operator under symmetric distance for fixed- and variable-sized multisets, which we apply here for crossover, and in the following section, based on the same metric, for mutation.

Note that due to the use of elitist $(\mu + \lambda)$ selection (see Section 6.1.6.2), which applies strong selection pressure, the crossover rate is fixed at $p_c = 1$. Thus in each generation no parents will be added to the offspring directly without breeding. In contrast, a rate $p_c = 0$ would mean that no offspring are generated, with the parents passing through unchanged to the set of offspring to undergo mutation and so on.

6.1.4.1 Multiset Operator

According to the geometric interpretation of crossover for multisets, given two candidate solutions, or multisets, $\psi_i = (C, m_C)$ and $\psi_j = (D, m_D)$, any crossover operator that returns offspring (O, f) such that

$$(C, m_C) \cap (D, m_D) \subseteq (O, f) \subseteq (C, m_C) \cup (D, m_D)$$

is considered a geometric crossover operator under symmetric distance (Moraglio and Poli, 2006). The operator used here is defined as follows.

To restrict the space of candidate solutions for the EA, the initialisation procedure, discussed in Section 6.1.2, generates candidate solutions that cover the total network supply, B, and satisfy the supply at each $t \in S$, where, as a reminder, $S = \{t \in [T] | b_s(t) > 0\}$ and $\sum_{t \in S} b_s(t) = B$. To maintain this aspect of candidate feasibility, the crossover operator is required to cross-over fixed-size multisets while maintaining these constraints (see 6.1.2 for more details). To fulfil these requirements, the operator is designed to generate 'sub-offspring' for each $t \in S$, given $b_s(t)$, and then combine them to form offspring with total cardinality B. The method for generating 'sub-offspring' is described first, and this is followed by the method for combining them to generate complete offspring.

For a chromosome $\psi_i = (C, m_C)$ define the submultiset $\psi_{i,t} = (C_t, m_{C_t})$, for $t \in S$, where $C_t = \{(\sigma, t') \in C | t' = t\}$ and $\forall c_t \in C_t$, $m_{C_t}(c_t) = m_C(c_t)$, where $m_C(c_t) > 0$, as the submultiset consisting of all those genes in $\psi_{i,t}$ with positive flow departing s at time t. For candidate solution $\psi_j = (D, m_D)$, we then also have $\psi_{j,t} = (D_t, m_{D_t})$, for $t \in S$.

To generate sub-offspring for $t \in S$, we are required to sample a multiset (O_t, f_t) such that:

$$(C_t, m_{C_t}) \cap (D_t, m_{D_t}) \subseteq (O_t, f_t) \subseteq (C_t, m_{C_t}) \cup (D_t, m_{D_t}) \tag{6.1}$$

The set of sub-offspring for $t \in S$, denoted by Θ_t, from which to sample the selected sub-offspring (O_t, f_t) , is defined as follows.

Let the symmetric difference of submultisets (C_t, m_{C_t}) and (D_t, m_{D_t}) for time $t \in S$ be $(C_t \Delta D_t, n_t)$, which is the multiset of the difference between their union and their intersection, as described in the preliminaries (6.1.3.1). Let the intersection between the submultisets for $t \in S$ be $(Q_t, n_{Q_t}) = (C_t, m_{C_t}) \cap (D_t, m_{D_t})$. Now consider the constraint that the potential sub-offspring $(O_t, f_t) \in \Theta_t$ must be a superset of the intersection of the parents (from Equation 6.1), where $(Q_t, n_{Q_t}) \subseteq (O_t, f_t)$. Given the supply at time $t \in S$, $b_s(t)$, it is necessary to restrict the cardinality of the potential sub-offspring, such that, given the set

$$\Upsilon_t = \{(U_t, m_{U_t}) \in \mathcal{P}\big((C_t \Delta D_t, n_t)\big) : |(U_t, m_{U_t})| = b_s(t) - |(Q_t, n_{Q_t})|\}$$

the set of potential sub-offspring is:

$$\Theta_t = \bigcup_{(U_t, m_{U_t}) \in \Upsilon_t} \{(Q_t, n_{Q_t}) \cup (U_t, m_{U_t})\}$$

Due to the constrained cardinality of the elements in Υ_t, the total cardinality of each $(O_t, f_t) \in \Theta_t$ is $|(O_t, f_t)| = b_s(t)$, which is required to satisfy the constraint that the generated offspring have a total cardinality B.

It follows that if the submultisets have no intersection: $(Q_t, n_{Q_t}) = \emptyset$, for $t \in S$, then $\Theta_t = \Upsilon_t$. Or, if $(C_t, m_{C_t}) = (D_t, m_{D_t})$, then $(Q_t, n_{Q_t}) = (C_t, m_{C_t}) = (D_t, m_{D_t})$ and hence $\Upsilon_t = \{\}$ and $\Theta_t = \{(Q_t, n_{Q_t})\}$, so that the number of sub-offspring is 1. In this case, the sub-offspring and the submultisets of the parents will be genetically identical, and if this holds for all $t \in S$, then the offspring will be identical to the parents.

To then generate the complete offspring (O, f) from the sub-offspring, we combine uniformly sampled sub-offspring from the sets Θ_t, for each $t \in S$, by taking the union of each (O_t, f_t):

$$(O, f) = \bigcup_{t \in S} (O_t, f_t)$$

To generate the set of offspring, two offspring are sampled for each pair of parents selected by the mating selection scheme.

6.1.5 Mutation

The goal of mutation is to introduce diversity into the population's gene pool. It works by moving a current candidate solution, from the newly generated offspring, to a new candidate solution. Specifically, this process is carried out by considering each gene of a chromosome for application of a move operator (defined below). The decision of whether to apply the operator to any particular gene is a stochastic process governed by a predefined mutation rate, $p_m \in [0,1]$. At the extremes, a rate $p_m = 0$ would mean no genes are mutated and the offspring are left unperturbed; conversely, a rate of $p_m = 1$ would mean that all genes of all offspring are mutated, which would ultimately have a destructive effect on the search process.

Mutation is here based on the uniform geometric mutation operator defined by Moraglio and Poli (2004) with application to fixed-size multisets.

6.1.5.1 Multiset Operator

The mutation operator is defined using the neighbourhood function, N, described in Section 6.1.3. The neighbourhood was defined using a single unit of symmetric distance between multisets as a metric, where, for ψ, $\forall\psi' \in N(\psi)$ we have $d(\psi,\psi') = 1$. Based on this, the mutation operator is defined as a function that perturbs a candidate solution ψ to $\psi' \in N(\psi)$ by approximately uniformly sampling the neighbourhood $N(\psi)$. Specifically, for each offspring, each of the n genes is sequentially considered for mutation. If a gene, (σ,t), is chosen for mutation, then σ is replaced by another path in the path-space, $\sigma' \in \Omega, \sigma' \neq \sigma$, generated by the path-finding procedure described in Section 5.1. Note that to maintain the feasibility of the supply structure, the t values are not perturbed.

Under these conditions, it is possible for the mutation operator to be applied to different genes within a candidate solution and for the net effect of these perturbations to be such that it ends up back as the original candidate; given a large enough solution space, which is the case here, this is unlikely.

6.1.6 Selection Strategies

As part of an EA, two selection strategies are defined: a mating selection scheme, for selecting individuals each generation from the current population for mating to produce offspring, and a replacement strategy, used to choose the new population based on the current population and the offspring.

6.1.6.1 Mating Selection

For mating selection, a random mating selection operator (Bäck et al., 1997) is employed, whereby individuals from the population are sampled uniformly randomly (with replacement) from the population to be used to generate offspring. Two parents are needed to produce two children, so that a total of $(\lambda = \mu)$ parents are selected.

As shown in Section 5.4.7, this operator adds no comparisons to the $\mathcal{OCBA}^{EA}_{\delta*}$'s comparison set $\mathcal{C}$, since the mating selection operator is not competitive (it is not based on the relative ranking of parents).

6.1.6.2 Replacement strategy

For replacement, a standard elitist $(\mu + \mu)$ (*Plus-selection*) EA replacement strategy (Bäck et al., 1997) is used to determine the next generation from the combined current population and the children. The elitist selection operator selects the μ best members of the population and hence allows only super-fit offspring to pass through to the next generation. Unlike some operators, it ensures that the best candidate solution (based on observed fitness values) from the combined current population and offspring for each generation is guaranteed to succeed to the next generation.

As described in Section 5.4.7, this operator adds the following pairs to the $\mathcal{OCBA}^{EA}_{\delta*}$'s comparison set $\mathcal{C}$:

$$\mathcal{C} \leftarrow \bigcup_{i=(\lambda=\mu)+1}^{\mu+(\lambda=\mu)} \{\langle(i),(1)\rangle,\ldots,\langle(i),(\lambda=\mu)\rangle\}$$

The new population is determined from the combined current population and offspring $\mathcal{P}_{k-1} \cup O$, again as detailed in Section 5.4.7, where the top μ candidate solutions are selected from the total ordering of the current population and the offspring:

$$\mathcal{P}_k = \{((\lambda=\mu)+1),\ldots,((\lambda=\mu)+\mu)\}$$

6.1.7 Evaluation

This subsection discusses the procedure for evaluating the fitness of candidate solutions in the context of the NEA. Firstly, it discusses the noisy fitness function used to evaluate the quality of chromosomes, and, secondly, it discusses the integrated $\mathcal{OCBA}^{EA}_{\delta*}$ approach.

6.1.7.1 Noisy Fitness Function

The goal of the fitness function, $f : \Psi \rightarrow \mathbb{R}$, is to provide a measure of the quality of candidate solutions for the MSE problem. For COPs, it is often the case that the objective function is used directly, e.g. $f = G$, and for SCOPs, where the exact evaluation of individuals is often unavailable, the fitness is commonly defined as a simulation approximation, e.g. a Monte Carlo estimate, of the exact objective function. As shown in Chapter 5, an exact evaluation of the solution quality of candidate solutions for the MSE is in general impractical, and therefore the option of utilising the objective function directly as the fitness function is unavailable. However, it is possible to define a noisy fitness function using the approximate objective function (Equation 5.1), so that

$$f_n = \hat{G}^n \tag{6.2}$$

This fitness function is called the "original noisy fitness function".

However, preliminary experiments indicate that due to the prevalence of poor quality, or simply infeasible, candidate solutions in initial populations, this approach is inadequate for the effective use of the EA. The EA appears to suffer from the oft-cited problem of generating many infeasible candidate solutions for constrained COPs, and many poor quality candidate solutions. A common approach for resolving this problem, which is employed here, is to modify the fitness function for handling problem constraints.

6.1.7.2 Enhanced Noisy Fitness Function

Many approaches have been suggested for handling constraints with EAs, such as penalty functions, repair algorithms, etc. (for overviews see Michalewicz (1997); Coello Coello (2002)). By far the most common approach is the use of *exterior penalty functions*[6] (Coello Coello, 2002) (herein referred to as 'penalty functions'). In general, penalty functions are used to penalise individuals by adjusting their fitness values depending on their constraint violations, with the aim of penalising those candidates with more violations.

The theory of penalty functions indicates that it is better to include information about the distance of individuals from the feasible region rather than, say, a count of the number of violations. Based on this, we introduce an enhanced fitness function that includes a measure of the distance from the infeasible region towards the feasible region over all network states[7]. Note that because the objective function of the MSE

[6]Exterior penalty functions start with an infeasible solution and attempt to move towards the feasible region. The fact that they do not require an initial feasible candidate solutions is one of their main advantages over *interior penalty functions*, which start with feasible candidates and make it undesirable (costly) to approach the constraint boundary.

[7]The enhanced fitness function is defined for network states, however, it could easily be defined for flow pattern states. This is discussed more later.

(Equation 4.5) considers feasibility directly, in other words the best candidate solution is exactly the one with the fewest constraint violations per state over all states, one approach is to relax the problem constraints and reward candidates according to how close they are to the feasible region, using, for example, the expected distance to feasibility. For the MSE, the ideal solution has an expected distance of 0 — it is feasible almost surely across the state distribution, and therefore has an expected distance of 0, almost surely. However, the expected value for a candidate solution is in general unavailable as it would require enumerating all states. Nevertheless, in a manner similar to the method described in Section 5.2, we now propose a method for efficiently approximating the expected value using the average distance from the feasible region of a set of i.i.d. sampled network states.

However, rather than adding an additional penalty term to the fitness function as is customary, the enhanced noisy fitness function is based directly on the general formulation for exterior penalty functions. This fitness function utilises information about constraint violations and provides an estimate of the expected distance from feasibility. For the MSE, this information is derived for each network state from flow conservation constraints (Equation 4.2), and non-negativity and capacity constraints (Equation 4.3).

As a reminder, note that the network model assumes $\forall t > T$ (i.e., beyond the time frame of interest) that the values for edge properties remain static (for more details, see Notation & Preliminaries in Section 4.1.1). Furthermore, the demand at the network sink, l, is:

$$b_l(t) = \begin{cases} 0 & t \in [T-1] \\ -B & t = T \\ 0 & t > T \end{cases}$$

Even if flow arrives before T, it implicitly waits until time T. Therefore, if the demand at the sink node is not satisfied by time T, the flow conservation constraint is violated. Also, since paths are assumed finite with a maximum number of nodes v, it is assumed that all flow will eventually reach the sink by some $T' = T + k, k \in \mathbb{N}$, such that $b_l(T') = -B$.

For the MSE problem, as previously mentioned, constraints include flow conservation constraints (Equation 4.2), and non-negativity and capacity constraints (Equation 4.3). For defining the fitness function here, we split these constraints into the two groups: equalities and inequalities, and we transform the constraints to a suitable form for use by the enhanced fitness function.

For the inequalities, there are a total of w constraints, split between capacity and non-negativity constraints, such that $w = w_1 + w_2$, where w_1 is the number of capacity constraints, and w_2 the number of non-negativity constraints.

For use with the penalty function, the capacity constraints (Equation 4.3) are converted to inequalities in the form:

$$x_{ij}(t) - \mu_{ij}(t) \leq 0, \forall (i,j) \in \mathscr{E}, \forall t \in [T+k]$$

which states that the flow along an edge should not exceed the capacity of that edge. We denote the left-hand side of each constraint, given a candidate solution ψ and network state ξ, by the function $g_k(\psi,\xi)$, where $k = 1,\dots,w_1$.

Similarly, the non-negativity constraints (Equation 4.3) are transformed to inequalities:

$$-x_{ij}(t) \leq 0, \forall (i,j) \in \mathscr{E}, \forall t \in [T+k]$$

which states that flow along an edge should not be negative. Again, we denote the left-hand side of the inequality by the function $g_k(\psi,\xi)$, with $k = w_1+1,\dots,w = w_1+w_2$.

All MSE equality constraints, of which there are a total of z, involve flow conservation constraints (Equation 4.2). These constraints are transformed into equality constraints in the form:

$$\Big(\sum_{j\in\Gamma^{+1}(i)} x_{ij}(t) - \sum_{j\in\Gamma^{-1}(i)} \sum_{\{t'|t'+\tau_{ji}(t')=t\}} x_{ji}(t') \Big) - b_i(t) = 0, \forall i \in \mathscr{V}, \forall t \in [T+k]$$

which states that the sum of the flow entering and that leaving a node should equal the supply/demand at the node. For a candidate solution ψ and network state ξ, we denote the left-hand side of the equation by the function $h_k(\psi,\xi)$, with $k = 1,\dots,z$.

Then based on the general formulation of the penalty function from Coello Coello (2002), for a candidate ψ and a network state ξ, the total distance from feasibility is given by the function $J : \Psi \times \Xi \to (-\infty, 0]$ where

$$J(\psi,\xi) = -\Big[\sum_{k=1}^{w} \max(0, g_k(\psi,\xi)) + \sum_{k=1}^{z} |h_k(\psi,\xi)| \Big]$$

If a candidate solution is feasible on a network state then $J(\psi,\xi) = 0$.

Then the expected distance from feasibility for a solution ψ is given by

$$\mathbb{E}[J(\psi,\bar{\xi})] = \sum_{\xi\in\Xi} J(\psi,\xi)\cdot\mathbb{P}(\xi)$$

If a candidate solution is feasible on all network states $\mathbb{E}[J(\psi,\bar{\xi})] = 0$, then by the objective function of the MSE problem (Equation 4.5), we have that $G(\psi) = 1$ (see Section 4.3).

Based on this, the enhanced fitness function is defined

$$\hat{f}(\psi) = \mathbb{E}[J(\psi,\bar{\xi})]$$

However, evaluating this value exactly would require a full enumeration of all network states, which in general is impractical (as discussed in 5.2). Therefore, the fitness value for a solution is estimated using a set of i.i.d sampled network pattern states $\xi^i, i = 1,\dots,n$ from Ξ according to $\mathbb{P}$, by the unbiased MC estimator:

$$\hat{f}_n(\psi) = \frac{1}{n}\sum_{i=1}^{n} J(\psi,\xi^i)$$

By the law of large numbers, estimates will converge to the expected value as $n \longrightarrow \infty$. The variance of $\hat{f}(\psi)$ can be estimated by the sample variance:

$$\hat{s}_n^2(\psi) = \frac{1}{n-1}\sum_{i=1}^{n}(J(\psi,\xi^i) - \hat{f}_n(\psi))^2$$

We term the function $\hat{f}_n$ as the "enhanced noisy fitness function", and the goal of the NEA is now to solve the proxy optimisation problem:

$$\max_{\psi\in\Psi} \hat{f}_n(\psi)$$

Note, also, that it is possible to concurrently use the i.i.d. sample of network states to evaluate the original noisy fitness function (Equation 6.2) value, f_n, since if $J(\psi,\xi) = 0$ then $I_{\mathcal{F}_\psi}(\xi) = 1$.

While we have defined the enhanced (noisy) fitness function for network states, we can analogously define it for flow pattern states. In fact, in practice we use it with flow pattern states, due to the same reasons as discussed in Section 5.2.4. The following algorithm describes how to compute the enhanced fitness function for a candidate solution for a single flow pattern state.

Based on the approach for sampling from the flow pattern state distribution (Algorithm 5.2.3), Algorithm 6.1.2 provides a high-level description of the procedure for evaluating the distance from feasibility for a single sampled flow pattern state.

The algorithm functions almost identically to Algorithm 5.2.3, except that rather than returning a sampled flow pattern state, it returns the distance from feasibility for a single sampled flow pattern state. Similar to Algorithm 5.2.3, the set Z maintains the current joint flow given the state realisation, through pairs of edges and departure times. Again, the set $\bar{R}_{ijt}$ stores the pairs of paths and departure times associated with the current joint flow for edge (i, j) at departure time t.

Algorithm 6.1.2: SAMPLESTATEDISTANCEFROMFEASIBILITY($\mathcal{N}, \psi = (R, m)$)

```
$c \leftarrow 0$
$\bar{R}_{ijt} \leftarrow \emptyset, \forall (i,j) \in \mathcal{E}, \forall t \in [T]$
$Z \leftarrow \{((i_1,i_2),t_0) | i_1 = s, i_2 \in \sigma, (\sigma,t_0) \in R, m((\sigma,t_0)) > 0\}$
$\bar{R}_{ijt} \leftarrow \{(\sigma,t_0) \in R | i = i_k, j = i_{k+1} \in \sigma, m((\sigma,t_0)) > 0\}, \forall ((i,j),t_0) \in Z$
while $Z \neq \emptyset$
do
    $((i,j),t) \leftarrow \arg\min_{((i',j'),t') \in Z} t'$
    if $t < T$
      then $t^* \leftarrow t$
      else $t^* \leftarrow T$
    $\tau \leftarrow$ SampleProbabilityDistribution($\tau_{ij}(t^*), \rho_{ij}(t^*)$)
    $t' \leftarrow t + \tau$
    if $j = l$ and $t' > T$
      then $c \leftarrow c - (t' - T)$
    $\mu \leftarrow$ SampleProbabilityDistribution($\mu_{ij}(t^*), \beta_{ij}(t^*)$)
    $\hat{x} \leftarrow \sum_{(\sigma,t_0) \in \bar{R}_{ijt}} m((\sigma,t_0))$
    $c \leftarrow c - \max(0, \hat{x} - \mu)$
    if $j \neq l$
      then
        $Z' \leftarrow \{((j,j'),t') | j = i_k, j' = i_{k+1} \in \sigma, (\sigma,t_0) \in \bar{R}_{ijt}\}$
        $\bar{R}_{jj't'} \leftarrow \bar{R}_{jj't'} \cup \{(\sigma,t_0) \in \bar{R}_{ijt} | j = i_k, j' = i_{k+1} \in \sigma\}, \forall ((j,j'),t') \in Z'$
        $Z \leftarrow Z \cup Z'$
    $Z \leftarrow Z \backslash \{((i,j),t)\}$
return $(c)$
```

The algorithm has similar worst-case computational complexity to Algorithm 5.2.3, $O(v^2T(D+H))$, except that the time frame of interest is now unbounded. The time horizon is unbounded since it will now evaluate (and penalise) infeasible solutions as well; however, the maximum length of a path is finite, and limited by the number of nodes in the network, v. In practice, it is necessary to put a limit on the unbounded time horizon, and in this work it is assumed that each edge's travel times are finite and less than T, so that worst-case complexity is $O(v^2(vT)(D+H)) = O(v^3T(D+H))$.

The enhanced (noisy) fitness function evaluates fitness values that enable the NEA to better discriminate between candidate solutions than the original (noisy) fitness function, in particular for poor quality candidates. It also allows the NEA to exploit potentially favourable adaptations that might be present in infeasible candidate solutions while directing the search towards the feasible region.

Note that an additional difficulty can occur due to the use of small sample sizes. A fitness estimate based on few samples may under or overestimate the solution quality of the individuals and ultimately reduce the effectiveness of the search, particularly for candidate solutions with high quality estimates. Therefore, to maintain competitiveness within the population additional samples, $\bar{n}_0$, are allocated each generation to individuals with fitness values $\hat{f}_n(\cdot) = 0$.

6.1.7.3 $\mathcal{OCBA}^{EA}_{\delta^*}$ for the MSE

As previously discussed, due to the uncertainty in the evaluation of each candidate solution, the statistical ranking procedure $\mathcal{OCBA}^{EA}_{\delta^*}$ (detailed in Section 5.4.9) is integrated with the NEA. The NEA that utilises the $\mathcal{OCBA}^{EA}_{\delta^*}$ procedure is called the NEA$_{OCBA}$. In this context, the ranking procedure is used to guarantee at some level of confidence that the order information required by the NEA is correct, ultimately aiming to reduce the effect of noisy fitness evaluations on the search process. The ranking method is used each generation, after the offspring have been generated by crossover and perturbed through mutation, and before the replacement operator is applied, as shown in Algorithm 6.1.1. Once the ranking procedure has completed the NEA cycle continues as normal.

The ranking procedure has a number of assumptions that must be met for its effective use. Two important assumptions to address are: one, that fitness samples for each candidate solution are independent and normally distributed, and, two, that equality of fitnesses does not occur.

The first assumption that the fitness values are independent and normally distributed is important for the effective use of the OCBA method. If this condition is not met could lead to potentially poor, or inaccurate, sample allocations, and hence reduce the efficacy of the procedure. Using the enhanced noisy fitness function, by appealing to the central limit theorem, the assumption of normality is approximately satisfied for large sample sizes. For the initial sample sizes of use here, ≥ 30, in early EA generations, the fitness values are generally well approximated by normal distributions, and better approximated by Student's t distributions due to longer tails (as shown, for example, by the Q-Q plots in Figure 6.1). However, as the population's quality increases, the individual fitness distributions begin to right-skew as the number of constraint violations decreases — the candidate solutions are of higher quality. In fact, the worst right-skew occurs for candidate solutions with feasibility almost surely, that is, those that have no constraint violations — the ideal solution for the MSE. However, to what extent the skew effects the performance of the ranking procedure is uncertain.

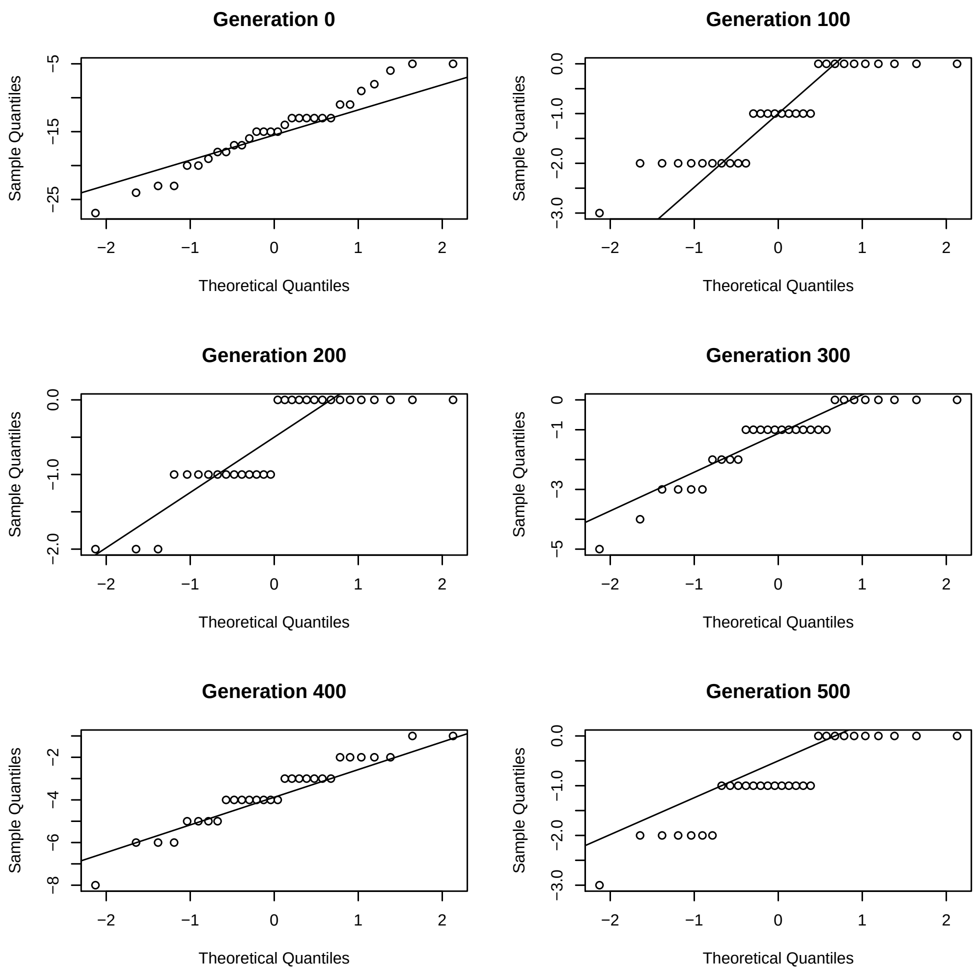


Figure 6.1: Normal Q-Q Plots of fitness distributions for a typical candidate solution in offspring population for respective generations. Taken from results of NEA$_{OCBA}$ for instance 13 of PC3 from the experimentation detailed in Section 7.2.3.

The second issue is whether equality of fitness values can be assumed not to occur. Two cases may arise that increase the likelihood of equality of fitnesses. The first is due to the fact that identical genotypes may exist within the population. Identical genotypes have equal expected fitness values, thus any variation in their estimated fitness is due to sampling variation, and as the population converges in later EA generations, these cases are more likely to occur. To prevent this from causing issues with the ranking procedure, when two genetically identical individuals exist in the population they are not compared using the statistical ranking procedure, that is, their respective comparison pair is not added to the set $\mathcal{C}$.

The second case occurs when different genotypes have the same expected fitness value. Again, in the later generations of an EA's run this is more likely to occur as candidate solutions become more genetically similar. However, unlike the previous case, in general it is impractical to determine whether two candidates with different genotypes have the same fitness value — this would require evaluating the expected fitness value. Therefore, no steps are taken to explicitly prevent this case arising, but it is assumed to be quite unlikely to occur in general.

An additional concern relates to the range of values for the enhanced noisy fitness function. This fitness function requires that the ranking procedure also works with candidate solutions that have estimated fitness values of 0 (and variance of 0). While additional samples are allocated to very fit individuals to maintain competitiveness in the population, this case may still arise, and since there is no measurable way to determine the uncertainty in the individual's sampled fitness, the proposed solution is to prune comparisons from $\mathcal{C}$ that involve these individuals. That is, for example, for an individual i' that has an observed fitness of 0, all comparisons involving i' are pruned, $\mathcal{C} = \mathcal{C} \setminus \{\langle i, j \rangle \in \mathcal{C} | i = i' \vee j = i'\}$, before the ranking procedure is executed. In general, this is more likely to occur in later EA generations due to the prevalence of good candidates; however, as previously mentioned, additional samples are allocated to individuals with high fitness values to reduce the chance of overestimating the quality of good candidate solutions.

6.1.8 Stopping Criteria

An additional requirement of an EA is stopping criteria, or termination conditions, that define the conditions under which the search process terminates. Typical criteria include the following, often used in combination: terminate once a maximum number of generations have passed; terminate on convergence, that is, stop if the population converges to a single fitness value or genotype; or terminate if the optimal solution has been discovered, assuming it can be identified. Of these three criteria, only the first is applicable, since in general optimal solutions cannot be identified for the MSE, and due to the use of a noisy fitness function, convergence to a single value is unlikely.

Furthermore, due to the need to compare fairly the NEA approach with a random search algorithm (described in Section 6.2), a simple, common termination criteria applicable to both algorithms is desired. Therefore a number of generations is chosen, κ, which can be utilised by the algorithms.

Determining an appropriate κ value for effective search is problem (instance) dependent, determined by, for example, the size and complexity of a particular instance. It is unlikely that a single value is suitable for all problem classes. For example, instances with a large search space may require a larger number of generations than those with small spaces. Suitable values for κ are discussed in Chapter 7.

6.1.8.1 Final Selection of Best

As described in Section 3.4.3 of the literature review, in this work EAs are seen as 'Optimisers', so that the result of single EA run is taken to be the best candidate of the final population. However, due to the uncertainty in the fitness evaluation of candidate solutions, there is no guarantee that the individual with the best observed fitness values is the true best of the final population. Therefore, at the end of the algorithm's run a statistical selection procedure, the OCBA for the selection of the best (discussed in Section 5.4.4), is employed to guarantee with a predefined level of confidence that the returned individual is the true best candidate of the final population.

As previously discussed (in Section 3.4.3), a more general approach to the problem of returning the best solution discovered by the search procedure is given by Boesel et al. (2003). In this case, the EA is seen as a 'Generator', not an 'Optimiser'. The proposed method requires storing all discovered candidate solutions and then running a post-search statistical selection method. Since this set of candidate solutions will most likely include many clearly inferior candidate solutions, these are filtered using a screening procedure before applying the selection method. We suggest that in practice it is unnecessary to include all points explored due to the prevalence of poor quality candidates at least in early iterations, and that in later generations individuals tend to be very similar or even identical, so that we focus only on the final population of the EA's run. Also, with the use of a high confidence level for the generational ranking procedure, specified by α^*, the probability that the best candidate based on observed fitness values survives each iteration is relatively high. This approach could easily be extended to, say, the last g generations; however, it is likely that the last few generations contain many of the same individuals and, furthermore, it is desired to keep the number of individuals relatively low for the selection procedure.

It is likely that the final population will consist of many genetically similar individuals, and the closer the quality of candidate solutions the greater the effort required to discriminate between them. Therefore, one approach to reduce the required effort for the selection is to prune individuals with the same evaluation; however, because evaluation is uncertain it is not possible to know whether two candidates have the same evaluation with certainty using finite resources. Another approach, given that the algorithm is expected to converge upon certain areas in the solution space, is based on the observation that a subset of those difficult comparisons are between chromosomes that are genetical identical: those that have the same genotype, but different estimated values due to the sampling process. Therefore, as before, genetically identical individuals are not compared against each other. For the selection, the OCBA procedure for the selection-of-the-best problem (detailed in 5.4.4) is executed in order to select the best candidate solution from the final population with a target confidence level. Furthermore, for the final selection, individuals are compared based on original noisy fitness function values (Equation 6.2), not fitness values using the enhanced noisy fitness function.

The set of parameters for the selection method are listed in Table 6.1. To evaluate an accurate estimate of the fitness of the individuals of the last population, a large initial sample size is utilised. A small indifference-zone is used to allow for good precision in the comparisons (see 5.2.2 for details on the issues of precision). Furthermore, a target confidence level $1-0.01 = 0.99$ is utilised to guarantee with a high level of confidence that the returned solution is actually the best of the final population. The remaining parameters, q and θ, were set given the recommendations discussed in Section 5.4.12.

Table 6.1: Final Selection Parameters

Parameter	Value	Description
$\bar{n}_0$	1000	initial sample size
q	10	number of individuals to select each stage out of k
θ	10	number of additional samples for each of q individuals
δ^*	0.01	indifference-zone
α^*	0.01	target confidence level: $1-\alpha^* = .99$

6.1.9 NEA$_{MEAN}$

To evaluate the effectiveness of the NEA$_{OCBA}$ procedure, specifically the $\mathcal{OCBA}^{EA}_{\delta^*}$ procedure for tackling noisy evaluations, we define an additional NEA approach, called the NEA$_{MEAN}$, that utilises a predefined, static resampling rate instead of a ranking procedure. The NEA$_{MEAN}$ approach takes a predetermined number of samples of the enhanced noisy fitness function for each individual and compares the candidate solutions based solely on their observed fitness values. In the literature this approach is known as explicit averaging of noise over time (as discussed in Section 3.4.2). The initialisation of the population is done using the same seeding as the NEA$_{OCBA}$, and the final selection procedure described above is also used.

The NEA$_{MEAN}$ utilises the same input parameters as the NEA$_{OCBA}$ procedure, where applicable.

6.1.10 NEA Parameters

The NEA approach has a number of input parameters, as shown in Table 6.2. For the integrated approach, NEA$_{OCBA}$, the $\mathcal{OCBA}^{EA}_{\delta^*}$ has additional parameters listed in Table 5.3, in Section 5.4.12.

Table 6.2: NEA Parameters

Parameter	Range	Comment
μ	$\mathbb{N}$	Population size
λ	$\mathbb{N}$	Offspring size, fixed $\lambda = \mu$
κ	$\mathbb{N}$	Generations
p_c	$[0,1]$	Crossover Rate, fixed $p_c = 1$
p_m	$[0,1]$	Mutation Rate

6.2 Random Search Algorithm

As described in the Section 4.4, it is possible to evaluate a lower bound on solution quality for the MSE by a point, or interval, estimate of any candidate solution for the MSE. However, a bound based on such an approach may not be particularly useful. In order to establish a more informative bound, and therefore provide a baseline reference for the NEA, a simple random search algorithm (RSA) is proposed. The results of the RSA provide a baseline for the NEA's performance and give some indication of the fitness distribution of the solution space.

In order to provide a fair baseline for the performance of the NEA, the RSA is designed identically to the NEA$_{OCBA}$ approach, except that the crossover and mutation operators are, effectively, disabled. The algorithm samples uniformly random candidate solutions, again using the procedure described in Section 5.1.3, from the search space and compares them using EA selection in order to find the current best, but it does not use the candidate solutions to bias the sampling in order to improve the search, as per the NEA, i.e. it performs no hill-climbing. In practice, to differentiate between the algorithms, for the RSA, the crossover and mutation rates are set to 0 and 1, respectively. Using these parameter values, the mating population will pass through to the offspring population without undergoing crossover — that is, no offspring are generated using the crossover operator; and they will then undergo full mutation, which is equivalent to generating random points in the solution space. Once again, no information is used from the current population in generation of the offspring. The offspring are then ranked against the current population of best candidates using the $\mathcal{OCBA}^{EA}_{\delta^*}$ procedure, as with the NEA$_{OCBA}$. The enhanced noisy fitness function of the NEA is used for evaluating candidate solutions, and, as before, a final selection is also run on the last population of the RSA. Given the complexities of evaluating and comparing solutions in stochastic environments, this approach allows for a fair comparison between the NEA approaches and the RSA, in particular highlighting the performance of the crossover and mutation operators.

6.2.1 Parameters

As with the NEA, a number of input parameters are required for the RSA. The parameters in common with the NEA are listed in Table 6.2. However, several parameter values differ from the NEA$_{OCBA}$ approach; these are given in Table 6.3.

Table 6.3: RSA Input Parameters that differ from NEA$_{OCBA}$

Parameter	Value	Comment
p_c	0	Crossover Rate
p_m	1	Mutation Rate

The approach also utilises identical parameter values for the final selection, as listed in Table 6.1.

6.3 Summary

In this chapter, given the complexity of the MSE problem in general, a metaheuristic approach based on a $(\mu + \lambda)$ EA combined with the $\mathcal{OCBA}_{\delta^*}$ procedure defined in the previous chapter (in Section 5.4.9), called the NEA$_{OCBA}$, was proposed for solving the MSE problem.

The chapter details the various components of the NEA approach and provides justification for several design decisions. The NEA comprises of the solution representation, the initialisation method, the neighbourhood function, the genetic operators: crossover and mutation, the selection schemes: mating and replacement strategies, the evaluation method, including the fitness function and the integrated ranking method, and, finally, the stopping criteria and final selection procedure.

To provide a comparison with the NEA$_{OCBA}$ approach, two additional procedures were also defined: an NEA variant, called the NEA$_{MEAN}$, that uses a predefined, static resampling rate for evaluating candidate solutions, and a simple random search algorithm, the RSA, that provides a baseline reference for the NEA approaches.

The following chapter details the experimentation carried out to evaluate the proposed approaches.

Chapter 7

Experimentation and Analysis

This chapter details the experimentation carried out for evaluating the proposed solution approaches for the MSE problem. It describes some implementation details, the experimental procedure, the preliminary experiments run for calibrating the approaches, and the main evaluation. Finally, it provides an analysis and discussion of the results, highlighting a number of emergent issues.

7.1 Implementation

All algorithms were implemented in Java and compiled using the Java 1.5 SDK from the Oracle Corporation[1]. Pseudorandom number generation within the solution approaches was implemented using the Mersenne Twister implementation of the CERN Colt library (version 1.2.0)[2]. All numerical computing, e.g. evaluating the Student's *t* cumulative distribution function, was also implemented using the Colt library.

7.1.1 Problem Instance Generator

For evaluating the proposed solution approaches, a problem instance generator was implemented to generate parameterised instances of the MSE due to the lack of an existing generator or reference instances for the MSE. We now describe a number of features of the generator, including the algorithms for generating topological graphs and network edge properties. Firstly, the topological graph is generated, then, secondly, the network is "populated" with network edge properties.

[1]Java can be found here: `http://www.java.com`.

[2]The Colt library can be found here: `http://acs.lbl.gov/software/colt/`.

The generation of problem instances is a pseudorandom process, and here instances are generated using pseudorandom numbers (seeds) provided by the Linux `/dev/urandom` utility[3].

7.1.1.1 Graph Topology

Similar to a series of works (Miller-Hooks and Patterson, 2004; Opasanon, 2004; Opasanon and Miller-Hooks, 2008; Miller-Hooks and Sorrel, 2008), the underlying graphs are parameterised by a number of nodes, v, and a maximum and minimum in- and out-degree for each node. The algorithm for generating the graphs is based on the procedure described by Newman (2003).

Briefly, for each graph to be generated, a random source, s, and sink, l, are selected, $s \neq l$, and a *degree sequence* is uniformly randomly generated. A degree sequence consists of a number of out-edges, chosen from within the range specified by the minimum and maximum node degree input parameters, for each node except the sink (which has no out-edges). Based on the degree sequence, pairs of nodes, i and j, are selected uniformly randomly and an edge (i, j) is added to the graph, and the degree count of node i is incremented, given the following conditions hold:

1. The head node of the edge is not the sink, $i \neq l$;
2. The target node is not the source, $j \neq s$;
3. No edge already exists between the nodes, $(i, j) \notin \mathscr{E}$;
4. The head node is not the source and the target node is not the sink, $i \neq s$ and $j \neq l$;
5. The head node's out-edge count is less than the value of the out-edges in the degree sequence;
6. The target node's in-edge count is less than the input maximum in-degree value.

For more details on these constraints, see the Notation and Preliminaries section (4.1). The process stops once the out-degree sequence has been satisfied for each node.

[3]`/dev/random` and `/dev/urandom` provide interfaces to the Linux kernel's random number generator. `/dev/urandom` is the unlocking counterpart to `/dev/random` that does not block to wait for a given level of entropy in the entropy pool. It is nevertheless regarded as safe for most cryptographic purposes, which, in general, require a high level of security. For more information, see, `http://www.kernel.org/doc/man-pages/online/pages/man4/random.4.html`

This method of generating random graphs for a given degree sequence generates graphs uniformly randomly from the ensemble of graphs for a particular degree sequence (Newman, 2003). Here the method samples graphs from all possible degree sequences for a given in- and out-degree range, and for all random source-sink pairs.

A further constraint on the graphs is that once generated, they are checked for $s-l$ connectedness, that is, that there must be at least one $s-l$ path in the graph. This is checked using a breadth-first graph search approach (Russell and Norvig, 2003). If no $s-l$ path is found, then there are no feasible solutions, and so a new topological graph is generated and checked again for connectedness, and so on. This method of generating topological graphs thus samples from the subset of graphs that have at least $s-l$ connectedness.

The problem classes defined for the experimentation, described in the rest of the chapter, concern graphs of 20 and 30 nodes; some average properties of the graphs generated for the experiments are given in Table 7.1.

Table 7.1: Measured Properties of Generated Graphs

	$n=20$		$n=30$	
Property	Mean	Std. Dev	Mean	Std. Dev
Number of edges	85.46	7.38	130.22	9.09
In- & out-degrees	4.5	0.39	4.49	0.31
Number of paths	$1,324,325.23$	$2,126,378$	$19,312,665.02$	$2,865,306$

7.1.1.2 Edge Properties

The edge properties for the STV Networks are also pseudorandomly selected given input ranges. Each network edge property, that is, travel times and capacities, for each time interval, $t \in [T]$, is modelled by a discrete probability distribution (for more details see Section 4.1). The number of elements in the distributions (the size of the supports), D, the number of capacity values, and H, the number of travel time values, and the range of these values are input as parameters. The number and range of values are assumed to be constant for all edges and time intervals. Furthermore, all distributions are assumed to be independent both over time and over space.

The method for generating random edge property probability distributions is based on that described in (Miller-Hooks and Mahmassani, 1998). To generate a distribution for a particular edge property for a specific time interval, a number of elements, either D or H, are uniformly picked without replacement from the permissible range of values for that property, and each is assigned a real number within the range $[0,1]$. The values are then normalised so that the sum of the real numbers (probabilities) is equal to 1. This procedure is executed for all edges over time for capacities and travel times.

7.2 Experimentation

The goal of the empirical investigation detailed here is to study the performance of the NEA proposed in the preceding chapter for finding solutions to the MSE problem. However, as explained previously, optimal solutions for the MSE problem are in general unavailable, and the complexity results of Section 4.3.3 suggest that it is unlikely that a general, efficient algorithm can be proposed for solving the MSE problem. Therefore, this section aims to investigate the claim that while the main approach, the NEA$_{OCBA}$, cannot guarantee optimal solutions, it can still find *useful* solutions, where useful solutions are those that are of reasonable quality and are found within reasonable time. These criteria are established by comparing the performance of the NEA$_{OCBA}$ against a baseline approach and upper bounds on optimal solutions for a number of sub-classes of the MSE problem. Upper bounds are established through the stochastic upper bounding method, the UB procedure, described in Section 4.4, and the baseline by the random search algorithm, the RSA, described in Section 6.2.

Furthermore, to the knowledge of the author, the combined EA and OCBA framework of Schmidt (2007), herein implemented (and extended with an optimisation and a heuristic) as the NEA$_{OCBA}$, has yet to be evaluated over a whole EA run, only for a single generation. Therefore, a further goal of the experimentation is to study the performance of the NEA$_{OCBA}$ over whole runs (generations > 1), investigating the effect of different algorithm parameterisations on performance. For investigating the influence of the OCBA method in particular, a NEA variant is also utilised, the NEA$_{MEAN}$ (described in Section 6.1.9), that uses a predefined resampling rate for tackling the noise. The NEA$_{MEAN}$ is used to compare against the NEA$_{OCBA}$ approach as a baseline method for handling noisy fitness functions, since it uses only the initial sample size for evaluating the fitness of individuals. To provide an additional baseline comparison with both NEA approaches, the RSA (described in Section 6.2) is also used.

The experimentation was carried out on a 16-core PC running Linux (kernel version 2.6.18) with 48GB RAM, where each CPU is an Intel Xeon 2.67GHz. Despite being a multi-core machine, each independent process was restricted to a single thread, did not utilise any parallel capabilities, and was limited to 2GB RAM. The seeding of algorithm pseudorandom number generators for experimentation was done using the `/dev/urandom` utility.

This next section details the design for the experimentation, including the classes of problem to be investigated, the performance criteria of interest, and the experimental procedure.

7.2.1 Experimental Design

The general experimental aim is to evaluate the algorithms on a number of different sub-classes of MSE problem, which we call a *problem class* and is defined by a set of parameter values that implicitly define a (possibly infinite) number of individual MSE problems. For each problem class, instances can be generated by the instance generator described above to which the algorithms can then be applied. The algorithms are compared based on their performance per problem class (that is, the performance of each algorithm on individual problems drawn from each of the problem classes provides some comparative indication of its overall "quality" for that class), where performance is defined in terms of solution quality, computational time, and the total number of fitness samples required.

The following subsections detail the problem classes and performance criteria of interest, followed by the experimental procedure for this investigation.

7.2.1.1 Problem Classes

Given the model detailed in the Notation and Preliminaries section (4.1), there are an infinite number of potential problem classes. For the purposes of experimentation, it is therefore necessary to restrict the problem classes to a small number of particular interest. The motivation for selecting a particular collection of classes could be, for example, because the real-world problem provides restrictions or constraints on the structure of problem instances, or perhaps in order to exclude trivial or over-constrained classes, e.g. problem classes for networks with, say, 2 nodes may not be informative. Furthermore, as is common in the study of algorithms, if algorithms perform well on worst-case instances, it is not unreasonable to expect them to perform well on best- or average-case instances.

The experimental design proposed herein is based on that of several works (Miller-

Hooks and Patterson, 2004; Opasanon, 2004; Opasanon and Miller-Hooks, 2008; Miller-Hooks and Sorrel, 2008), which describe the application of network flow and metaheuristic algorithms on a number of flow problems (as described in the literature review). In particular, Opasanon (2004) and Miller-Hooks and Sorrel (2008) evaluated metaheuristic approaches on similar hard flow problems in STV Networks. In common with these works, here the problem classes of interest are chosen in order to generate non-trivial and reasonably hard problem instances (although not so hard as to require more time and resources than are available for experimentation). The goal is to demonstrate the value of the proposed algorithms in finding useful solutions for these instances.

The specific restrictions on parameters are based on the domain of interest: emergency movement in the built environment. As discussed in the literature review, Section 2.4, for this domain, network models represent building circulation systems, and the particular optimisation problem of interest represents the task of planning efficient emergency movement in the built environment during, for example, a fire hazard scenario. An example restriction, adopted here, is that usually in a building the connectivity of particular points, modelled by nodes, e.g. doors and intersections between corridors, is low, and hence the connectivity of nodes for the generated graphs is also low. Also, the availability of supply is limited to model the movement of units at the beginning of the time horizon (as described in (SFPE, 2002)), for example, by taking $t = 0$ as the time an alarm sounds. However, since units often do not immediately begin evacuating, the time of departure of supply is modelled by the range $t \in [0,2]$. For edges, travel times are modelled using a range of possible values to represent the notion that some edges require a short time to traverse while others take longer, and capacities are limited to suggest that in general passageways, e.g. corridors and stairwells, may have varying entry capacity. Note that here a capacity value of 0 is included in the range of values to model the possibility of untenable conditions, that is, that particular sections of the building may not be accessible or tenable for use.

Given that the generation of problem instances involves some randomisation, in particular the construction of the underlying topological graphs, for each problem class a huge number of distinct networks can be generated. Due to this freedom, the variation in the underlying graph structures effectively means that to perform well in general, algorithms cannot rely on (and hence exploit) the existence of specific problem structures. In reality, even considering the restrictions above, a building may only have one or two stairwells, and therefore it is likely to have substantially fewer feasible paths than any randomly generated instances. Still, these generated instances provide a convenient first stop for evaluating new algorithms, and in a sense represent worst-case scenarios. Nevertheless, future work should investigate structured networks, since these networks may exhibit common or exploitable network properties (that is, common characteristics of buildings), in particular the structure of the topological graph. This is discussed in more detail in the Future Work section (8.4) of Chapter 8.

Based on these considerations, for the experimental work described herein, a number of the network model parameters' values or ranges were fixed, as listed in Table 7.2. The parameter values that are not fixed are listed in Table 7.3. The chosen values aim to give a fair representation of potential problem classes, but also represent some trade-off with the computational effort required to solve instances. The purpose of this experimentation is to explore the potential of the proposed solution approaches. For specific applications, further experimentation is necessary, and this is discussed in the Future Work Section 8.4. Then the set of all problem classes for evaluation is given by the product of the possible values for each parameter in Table 7.3. The total number of problem classes is $2^3 = 8$, with parameter values and corresponding labels, as shown in Table 7.4.

Table 7.2: Fixed Problem Class Parameters

Parameters	Values/Ranges	Comments
$\mathbb{E}[\lvert\Gamma^{+1/-1}(\cdot)\rvert]$	$\mathbb{E}[2,7] = 4.5$	Expected in- and out-degree
$S = \{t \in [T] \mid b_s(t) > 0\}$	{0,1,2}	Supply availability
D	2	Number of capacity values
H	2	Number of travel times values
$\mu_{ij}^{d}(t)$	$[0,20]$	Edge capacity values
$\tau_{ij}^{h}(t)$	$[1,15]$	Edge travel time values

Table 7.3: Variable Problem Class Parameters

Parameters	Values	Comments
v	$\{20,30\}$	number of nodes
T	$\{300,600\}$	peak time horizon
$b_s(t)$	$\{5,10\}$	supply at $t \in \{0,1,2\}$

Table 7.4: Problem Classes

Problem Class	n	T	$b_s(t)$
PC1	20	300	5
PC2	20	300	10
PC3	20	600	5
PC4	20	600	10
PC5	30	300	5
PC6	30	300	10
PC7	30	600	5
PC8	30	600	10

7.2.1.2 Performance Criteria

To judge the performance of the approaches, two criteria are of particular interest: the quality of the solutions found and the cost of finding them. Due to stochastic elements in both the generation of problem instances and the NEAs, to evaluate the performance of algorithms for a problem class we look to the expected solution quality given the MSE's problem objective (Equation 4.5) and cost for different configurations of the algorithms on each problem class. In general, however, it is impractical to evaluate over all possible problem instances for a problem class, and over all realisations of an algorithm. In this case, the expected performance is estimated using a sample of problem instances and algorithm runs.

For solution quality, for the EA-based approaches, the result of a single execution is judged by the best observed candidate solution of the final generation. Since the approaches are stochastic algorithms, independent runs on the same instance are expected to produce different results. In this case, the sample average over a number of runs on a problem instance can be used to assess the performance of an algorithm. For the stochastic algorithms using noisy fitness functions, the final result returned by a single run is assumed to have a low, if not negligible, uncertainty in the evaluation value of the individual. This is reasonable since the returned result is obtained from a final selection procedure that utilises a high initial sample size ($\bar{n}_0 = 1000$) and a high target confidence level ($\alpha^* = 0.01$), as shown in Table 6.1; the assumption of low uncertainty is therefore taken to be satisfactory for the purposes of analysis.

For measuring the cost of finding the returned solution, computation time and the total number of fitness samples evaluated are used to gauge the required effort. For the EA-based approaches, the computation time of the algorithm is taken to include the initialisation of the population, the EA loop and the final selection. It does not include however, I/O times, that is, time for reading in problem instances. However, due to known limitations of using computation time as a performance metric[4], to provide a more balanced view we measure both the computational time of each independent run of the algorithm and the total number of fitness function samples. The total number of fitness samples consists of a count of all samples of individual fitness distributions, including those sampled during the final selection. Although fitness sampling is not the only computationally demanding part of the algorithms, e.g. initialisation of the population can also require a nonnegligible effort, over a whole EA run it dominates. Therefore, we expect computation time to be positively correlated with the total number of samples.

For this two-level experimental approach, where we have a distribution of problem instances per class and a number of stochastic algorithms, a common procedural question is: 'How many instances, how many runs?'. In other words, for a fixed amount of computation, where should the effort be spent in order to get a good estimate of the expected performance of an algorithm: investigating more problem instances, or on more algorithm replications per instance? Birattari (2009) investigated this question and proved that the minimum variance estimator for estimating the expected cost in this scenario for a fixed number of executions, k, is a single run of a metaheuristic on k problem class instances. This approach is adopted here, and a simple validation of the results of this approach is reported in Section 7.2.3.3.

7.2.1.3 Experimental Procedure

The sampling of instances of a problem class is a stochastic process due to the inclusion of stochastic parameters in the model. Therefore, in order to gauge the expected performance of algorithms across a class, the algorithms were evaluated on a set of i.i.d. sampled instances, termed meta-replications. Furthermore, due to Birattari, only a single algorithm replication was executed on each problem instance. The parameters for the experimental procedure are listed in Table 7.5.

[4]Computation time does not provide a wholly satisfactory objective performance measure because in practice it can be influenced by a number of factors, including, but not limited to, programming skill, hardware and processor load.

Table 7.5: Experimental Parameters

Parameter	Description
R_{PC}	Number of instances per class (meta-replications)
R_{Alg}	Number of algorithm runs per instance (replications)

7.2.2 Parameter Tuning

Before the main experimentation, the tuning of the parameters of the main algorithms was carried out in order to find parameter configurations that would lead to robust algorithm performance, where robust is taken to mean finding reasonable quality solutions in reasonable time. Here the solution approaches each require a number of input parameters, which are shown in Table 7.6, where the $\times$ symbol indicates that the parameter is used by the algorithm. Since the RSA and NEA$_{MEAN}$ are used to provide a baseline reference for the NEA$_{OCBA}$'s performance, the parameters are tuned for the latter and then where appropriate the same values are used by the competing algorithms.

Table 7.6: Algorithm Input Parameters

Parameter	Description	RSA	NEA$_{MEAN}$	NEA$_{OCBA}$
μ	Population size	$\times$	$\times$	$\times$
λ	Offspring size	$\times$	$\times$	$\times$
κ	Number of Generations	$\times$	$\times$	$\times$
p_c	Crossover Rate	$\times$	$\times$	$\times$
p_m	Mutation Rate	$\times$	$\times$	$\times$
$\bar{n}_0$	Initial sample size	$\times$	$\times$	$\times$
q	Individuals for allocation	$\times$		$\times$
θ	Additional sample size	$\times$		$\times$
δ^*	Indifference-zone	$\times$		$\times$
α^*	Significance level	$\times$		$\times$

7.2.2.1 Experimental Procedure

For the parameter tuning, a number of configurations were evaluated in order to choose the parameter set that leads to robust algorithm performance. In order to reduce the effort required and amplify the difference between configurations, the parameter configurations were evaluated on the same set of independently generated problem instances per problem class. Furthermore, given the number of parameter configurations and the computation time required to run the algorithms, the evaluation samples only a small number of problem instances. Due to this small sample size, the results are thus only indicative of the best configuration. To gauge the performance of the configurations, the stochastic upper bounding procedure was also used, with a sample size of 200. The set of values for parameters for the experimental procedure are shown in Table 7.7. In other words, for each problem class, 10 problem instances would be generated, with each algorithm run once on each instance.

Table 7.7: Experimental Parameters for Parameter Tuning

Parameter	Value
R_{PC}	10
R_{Alg}	1

7.2.2.2 NEA$_{OCBA}$ Parameters

The NEA$_{OCBA}$ procedure has a number of input parameters that are split between those for the NEA and those for the OCBA procedure, as previously discussed (in Section 6.1.10). Due to the large number of parameters, in order to restrict the set of potential parameter values a number of pilot studies were carried out to explore viable ranges of values. Once reasonable ranges for values were discovered, an in-depth study was carried out on a selected set of parameter configurations for the problem classes of interest.

For practical purposes, to reduce the space of parameters in the subsequent parameter tuning, a number of NEA parameters were fixed; these values are shown in Table 7.8. The values for the EA were fixed as discussed in Chapter 6. The OCBA parameters were fixed based on recommendations from the literature, as detailed in Section 5.4.12. The results of the pilot studies are shown in Table 7.9.

Table 7.8: Fixed Algorithm Input Parameters

Parameters	Value	Comments
λ	μ	number of offspring
p_c	1	crossover rate
q	10	individuals to select each OCBA stage
θ	10	additional samples for each of the q individuals

Table 7.9: Results from Pilot Studies

Parameters	Values	Comments on values
μ	≤ 50	poor solution quality
κ	≤ 200	poor solution quality
p_m	0.05	robust
$\bar{n}_0$	≤ 20	poor estimates; poor budget allocation
$\bar{n}_0$	$500, 1000$	oversampling; much effort spent on non-promising solutions
δ^*	$0.01, 0.05, 0.1$	significant effort spent on non-promising solutions

Given the results of the pilot studies, a set of parameter values were chosen for further investigation in order to find robust configurations across problem classes and also to study the algorithm under different parameterisations. These values are listed in Table 7.10. The possible variations in these values lead to a total of eight different configurations for the parameter tuning phase; these configurations for the NEA$_{OCBA}$ are shown in Table 7.11. The total number of experiments was: $|PC| \cdot R_{PC} \cdot R_{Alg} \cdot C = 8 \cdot 10 \cdot 1 \cdot 8 = 640$, where C is the number of configurations, and the total number of generated problem instances was: $|PC| \cdot R_{PC} = 8 \cdot 10 = 80$.

7.2.2.3 Results and Analysis

As previously discussed, the focus of this phase of the evaluation is to identify parameter configurations that lead to robust algorithm performance. In order to assess algorithm performance three performance criteria have been nominated: solution quality, computation time and total number of fitness samples, with solution quality having priority, followed by computation time, and then the total number of fitness samples.

Table 7.10: NEA$_{OCBA}$ Parameter Values for Parameter Tuning

Parameter	Value(s)
μ	100
κ	500
p_m	0.05
$\bar{n}_0$	$\{30, 50\}$
δ^*	$\{0.5, 1\}$
α^*	$\{.5, .05\}$

Table 7.11: NEA$_{OCBA}$ Configurations for Parameter Tuning

Configuration	μ	λ	κ	p_c	p_m	$\bar{n}_0$	q	θ	δ^*	α^*
C1	100	100	500	1	0.05	30	10	10	0.5	0.5
C2	100	100	500	1	0.05	30	10	10	0.5	0.05
C3	100	100	500	1	0.05	30	10	10	1	0.5
C4	100	100	500	1	0.05	30	10	10	1	0.05
C5	100	100	500	1	0.05	50	10	10	0.5	0.5
C6	100	100	500	1	0.05	50	10	10	0.5	0.05
C7	100	100	500	1	0.05	50	10	10	1	0.5
C8	100	100	500	1	0.05	50	10	10	1	0.05

The boxplots[5] in Figures 7.1 and 7.2 show the distributions of the results for solution quality, Figure 7.3 for computation time, and 7.4 for the total number of fitness samples. The UB was also used to provide an upper bound on the performance per problem class. Tables of the results for each problem instance per problem class, and for each algorithm configuration and performance criterion are provided in Tables A.1, A.2, A.3, A.4, A.5 and A.6 in Section A.1.

For analysis, due to the use of small sample sizes and the fact that the data was

[5]All boxplots were generated using the boxplot function of R (v. 2.13.0) of the R Project for Statistical Computing: `http://www.r-project.org/`. The boxes signify the interquartile range (IQR): Q3-Q1, that is, half of the sample values, with the thick line inside the box indicating the median value. The whiskers indicate the range of the values with the ends marking the minimum and maximum values, unless outliers (represented by empty circles) are present. In this case they indicate 1.5 IQR. The function plots outliers if points are outside 1.5 IQR of the upper or lower quartile. The sample mean is indicated using a triangle.

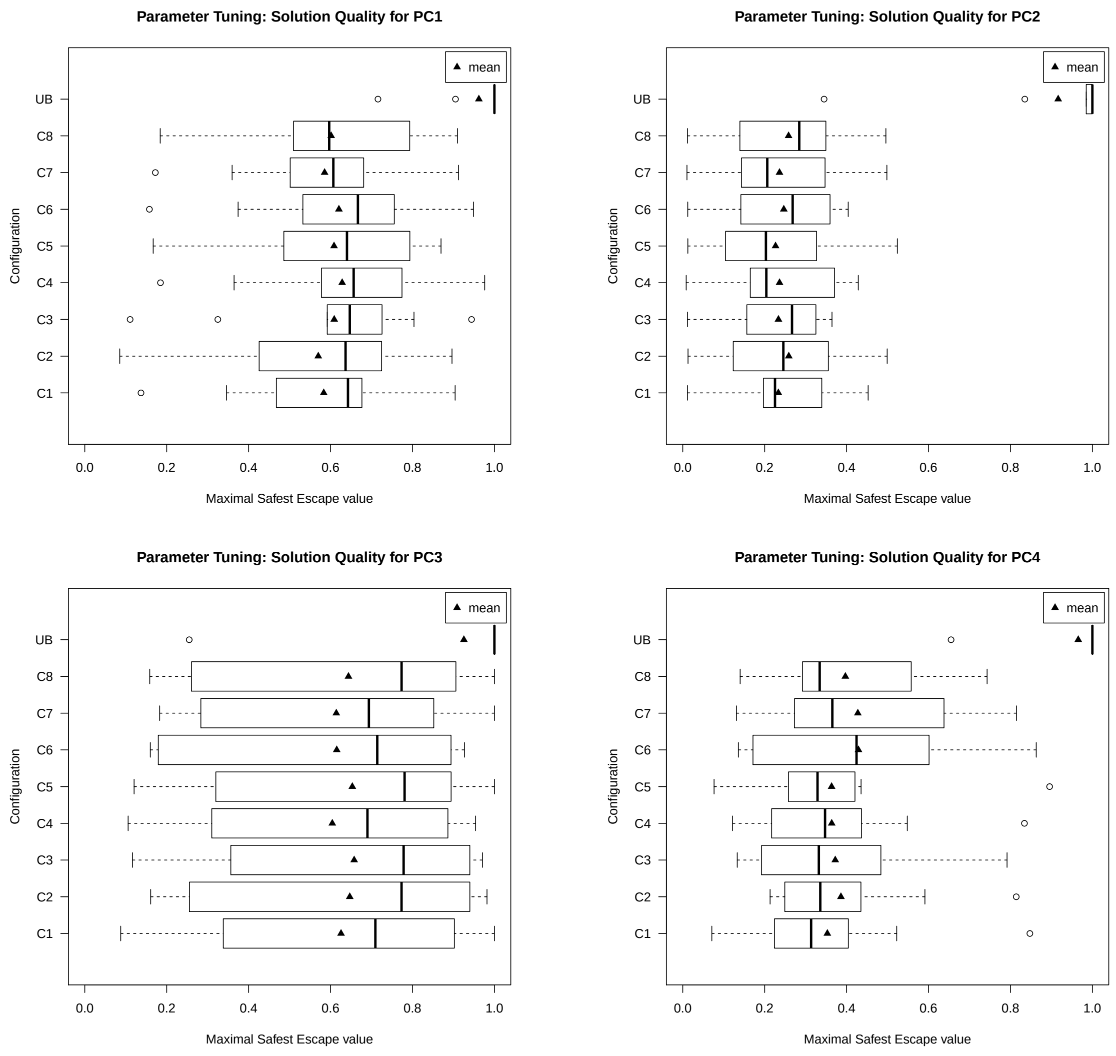


Figure 7.1: Parameter Tuning: SQ for PC1-PC4, C1-C8, UB

found to be unsuitable for parametric analysis[6], nonparametric statistical methods were used to study differences between the configurations. In particular, for this phase of the experimentation, Friedman's Two-Way Analysis of Variance by Ranks Test (Hollander and Wolfe, 1999) was employed to evaluate whether any differences exist between the configurations for the performance criteria and problem classes[7].

In terms of solution quality, the statistical tests indicated that over all problem classes there were no significant differences between the configurations at the 0.05

[6]Q-Q plots and the Shapiro-Wilk normality test were used to determine that the data was unsuitable for parametric analysis. Overall, the results were found not to approximate the normal distribution to a satisfactory level.

[7]All statistical tests for analysis were carried out using R functions included in the `stats`, `coin` and `multcomp` packages.

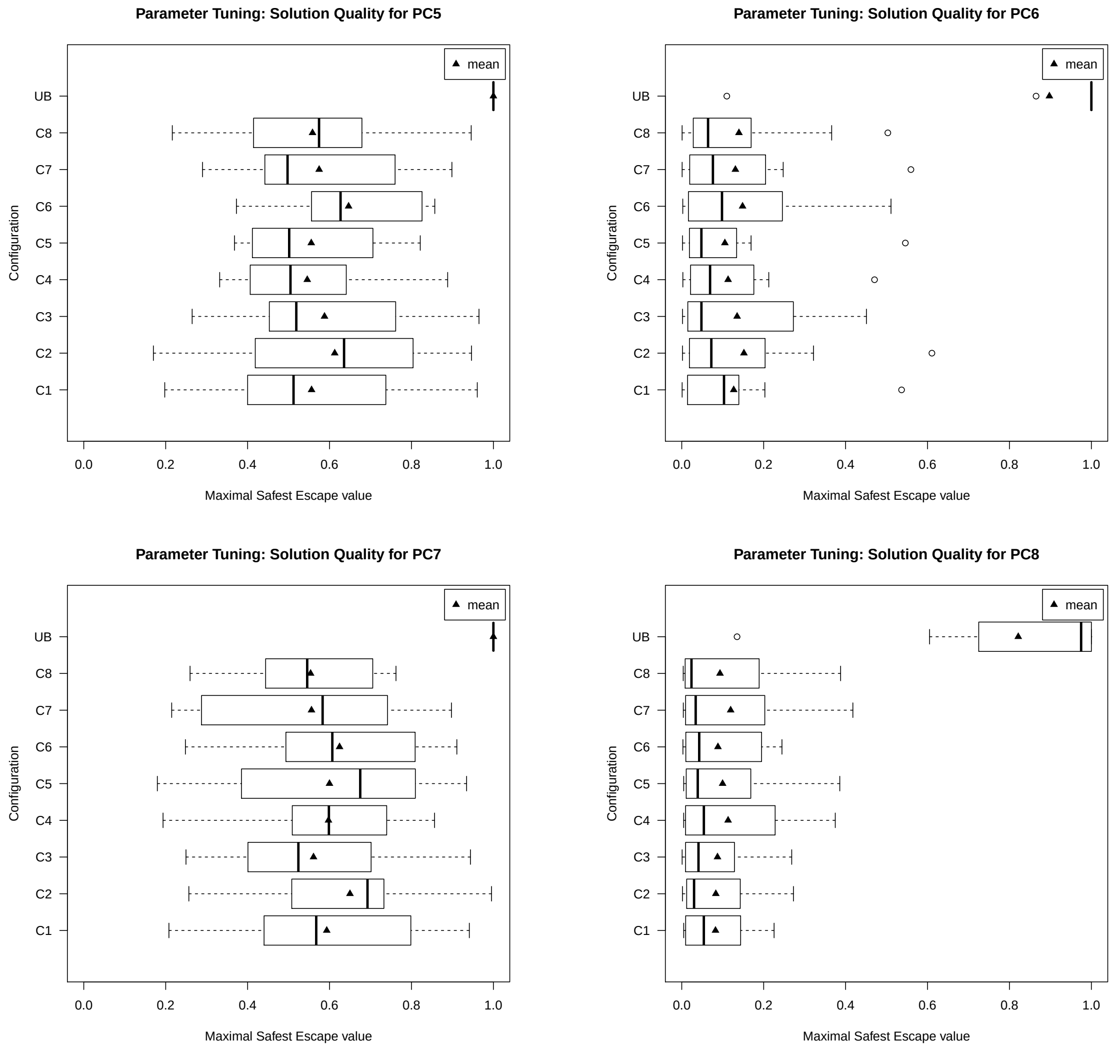


Figure 7.2: Parameter Tuning: SQ for PC5-PC8, C1-C8, UB

significance level. Across classes, however, as visible in Figures 7.1 and 7.2, there appears to be a trend that higher solution values are found for odd-numbered problem classes, i.e. PC1, PC3, PC5 and PC7; that is, those problem classes with lower supply. Furthermore, the data also suggest that PC1 and PC3 are the “easiest” of the classes (average solution quality > 0.6), and that PC6 and PC8 are the “hardest” (average solution quality $< .15$).

While there are no significant differences between configurations based on solution quality, there was variation in the cost of achieving the results. For computation time and total number of samples, Friedman’s Test was used again, and the null hypothesis was rejected at the 0.05 significance level for all problem classes. To inspect the differences, post hoc analysis was carried out using the two-sided Wilcoxon-Nemenyi-

McDonald-Thompson Test with p-value adjustment for multiple comparisons (Hollander and Wolfe, 1999) and an experimentwise error rate of 0.05[8]. Boxplots showing pairwise differences for computation time and number of fitness samples for each problem class are provided in Figures A.4 and A.5 of Appendix A.

For computation time and total number of fitness samples, of the three OCBA parameters, the choice of initial sample size, $\bar{n}_0 \in \{30, 50\}$, appears to have least effect on each criterion. However, the value of the indifference-zone does appear to have an impact on the required computational effort. Configurations utilising the smaller indifference-zone ($\delta^* = 0.5$), in particular C1, C2, C5, and C6, require, overall, significantly more computation time and number of samples than configurations using the larger zone ($\delta^* = 1$), that is, configurations C3, C4, C7, and C8. However, there appears to be no significant gain in solution quality for this additional effort. For target confidence levels, overall, the higher target level ($\alpha^* = 0.05$) does not appear to require significantly more effort than the lower value (0.5), despite the boxplots suggesting a difference[9]. Overall, the combined smaller indifference-zone and higher confidence target level appears to require the most computational effort, that is, configurations C2 and C6; but, again, for no gain in solution quality.

Based on these results, configurations C3 and C4 provide the best trade-off between the achieved solution quality and the effort required to obtain it. Neither of the configurations requires significantly more computation time or number of samples than other configurations across all problem classes, including between themselves. Of the two, C4 is chosen over C3, since it uses a higher target confidence level and thus provides more certainty in the ordering of individuals for no significant additional cost.

Building on these results, and in order to find parameters that improve solution quality, an additional three configurations using OCBA parameter values from C4 were evaluated to investigate the effect of more generations and larger populations. These configurations are provided in Table 7.12. The boxplots from these experiments, including C4, are shown in Figures 7.5 and 7.6, 7.7, and 7.8. The results are provided in Tables A.7 and A.8 for solution quality, in Tables A.9 and A.10 for computation time, and in Tables A.11 and A.12 for number of samples, in Appendix A.

Analysis of the results was once again done using Friedman's Test, and for comparison the C4 configuration was included in the tests. For each performance criterion and

[8]Friedman's and Wilcoxon-Nemenyi-McDonald-Thompson Tests were run using R code from Galili (2010), last accessed 1st of May 2011.

[9]Note that the boxplots do not consider the adjusted p-values required for the use of multiple comparison testing. For more on multiple comparison issues, see, for example, (Hsu, 1996).

Table 7.12: Additional NEA$_{OCBA}$ Configurations for Parameter Tuning (bold values indicate different values from C4)

Configuration	μ	λ	κ	p_c	p_m	$\bar{n}_0$	q	θ	δ^*	α^*
C9	100	100	**1000**	1	0.05	30	10	10	1	0.05
C10	**200**	**200**	500	1	0.05	30	10	10	1	0.05
C11	**200**	**200**	**1000**	1	0.05	30	10	10	1	0.05

each problem class, the null hypothesis was rejected at the 0.05 significance level, and so post hoc analysis was carried out again using the two-sided Wilcoxon-Nemenyi-McDonald-Thompson Test with p-value adjustment for multiple comparisons and an experimentwise error rate of 0.05. Boxplots showing the pairwise differences for each performance criterion and problem class are provided in Figure A.3 for solution quality, in Figure A.4 for computation time, and Figure A.5 for total number of fitness samples, in Appendix A.

The tests indicated that there was no advantage to using extra generations for a fixed population size, that is, there was no significant difference between C4 and C9, and C10 and C11, for solution quality or computational effort. For larger population sizes, however, there were significant differences. Configuration C11 had significant differences with C4 and C9 over all classes both for solution quality and computational effort. Configuration C10 had significant differences over C4 for all problem classes for solution quality and computation effort, except for solution quality for PC1. C10 also had significant differences in solution quality over C9 for the classes PC6, PC7 and PC8; yet did not require significantly more effort on any problem class.

Overall, the tests indicate that larger populations improve solution quality, as highlighted by the plots in Figures 7.5 and 7.6. However, as expected, this improvement can require significant additional effort, in particular for the "harder" classes, as shown in Figures 7.7 and 7.8. On the other hand, using more generations does not necessarily significantly improve solution quality, as seen from comparing the results of C4 with C9, and C10 with C11.

Based on these results, C10 was chosen as the configuration providing the best trade-off between solution quality and computation time over all problem classes. In terms of solution quality, the tests indicate that across all problem classes, C11 did not provide significantly higher values, and that for harder classes, the larger population was beneficial. For computational effort, it was never the case that C10 required significantly more effort than C9 or less than C11. In summary, there does not appear to be a significant advantage in choosing C11 over C10, but in general C10 provides a gain over C4 and C9.

The results for parameter tuning suggest that the NEA parameters have more impact on the overall performance of the algorithm than the OCBA parameters. This observation appears justified since NEA parameters used here control the extent to which the search space is explored, and the OCBA parameters effect the performance of the OCBA procedure and in turn determine how well the algorithm can exploit what it has discovered in order to guide the search, or, in other words, how mislead it is by error in fitness estimations. This point is further addressed in the Discussion section (7.4.1).

Finally, note that the selected parameter values are not necessarily optimal; more extensive experimentation would be required to establish such values. These results provide an indication of the behaviour of the main algorithm under varying parameter values, and contribute to the goal of demonstrating that the NEA$_{OCBA}$ framework can find high-quality solutions for the MSE in reasonable time; and are thus satisfactory for the purposes of this work.

7.2.3 Evaluation

The goal of this phase is to evaluate the performance of the NEA$_{OCBA}$ and NEA$_{MEAN}$ approaches for solving the MSE. To judge their performance, the algorithms are evaluated against a baseline approach and upper bounds on optimal solutions. The RSA is used to establish a baseline and upper bounds by the UB procedure.

7.2.3.1 Experimental Procedure

The experimental procedure for this phase follows closely the procedure used for parameter tuning. As before, a number of problem instances are generated for each of the problem classes (listed in Table 7.4) and the algorithms are evaluated on each instance. Here, however, problem instances are generated independently for each algorithm: NEA$_{OCBA}$, NEA$_{MEAN}$, RSA and UB. The experimental parameters for this phase are shown in Table 7.13; in particular, more problem instances are used than for the parameter tuning. The same performance criteria are of interest, and, as previously discussed, the algorithms share parameter values where appropriate.

The final parameter sets for the NEA$_{OCBA}$, NEA$_{MEAN}$ and RSA are shown in Table 7.14. The UB used a sample size of 200. The total number of algorithm runs was: $|PC| \cdot R_{PC} \cdot R_{Alg} \cdot A = 8 \cdot 30 \cdot 1 \cdot 4 = 960$, where A is the number of algorithms, and the number of independently generated problem instances was also 960.

Table 7.13: Experimental Parameters for Evaluation

Parameter	Value
R_{PC}	30
R_{Alg}	1

Table 7.14: Algorithm Input Parameters for all Problem Classes (bold values are not shared)

Parameter	RSA	NEA$_{MEAN}$	NEA$_{OCBA}$
μ	200	200	200
λ	200	200	200
κ	500	500	500
p_c	**0**	1	1
p_m	**1**	0.05	0.05
$\bar{n}_0$	30	30	30
q	10		10
θ	10		10
δ^*	1		1
α^*	0.05		0.05

7.2.3.2 Results and Analysis

The boxplots for the results of the experiments are shown in Figures 7.9 and 7.10 for solution quality, in Figure 7.11 for computation time, and in Figure 7.12 for total number of samples. Tables of results can be found in Section A.2 of Appendix A.

Once again nonparametric statistical methods were used for analysis because the data was judged to have failed to meet the assumptions for parametric methods. Nonparametric hypothesis tests were carried out to determine if the algorithms' results for each problem class and each performance criterion were significantly different. Since the algorithms were evaluated on independent samples for each problem class, the Kruskal-Wallis Test for general significance (Hollander and Wolfe, 1999) was employed with the null hypothesis that no significant difference exists between the algorithms for a given performance criterion and problem class. The null hypothesis was rejected for all performance criteria and problem classes at the 0.05 significance level; in other words, the differences between the algorithms were judged significant. In order to investigate individual differences, post hoc pairwise comparisons were evaluated using the two-sided Wilcoxon Rank Sum Test (Hollander and Wolfe, 1999) with Holm-Bonferroni correction for multiple comparisons (Holm, 1979) and an experimentwise error rate of 0.05. The p-values for the pairwise comparison tests are provided in Section A.2, in Tables A.19 and A.20 for solution quality, Tables A.21 and A.22 for computation time, and Table A.23 for total number of samples.

For solution quality, the tests showed that the distribution of values produced by the UB are significantly higher than the other three algorithms across all problem classes, while the RSA performs worst across all problem classes. For all classes, except PC1, there were no significant differences between the NEA$_{OCBA}$ and the NEA$_{MEAN}$; for PC1, the NEA$_{OCBA}$ performed better.

In terms of computational effort, evaluating the UB requires significantly less effort than the other three algorithms, and the NEA$_{MEAN}$ requires significantly less than the RSA and NEA$_{OCBA}$ for all problem classes. The NEA$_{OCBA}$ requires significantly less effort than the RSA, except for PC6, the “hardest” class, where it requires more.

For the number of fitness samples, the NEA$_{MEAN}$ uses significantly fewer fitness samples than the RSA and the NEA$_{OCBA}$. Comparing the RSA and the NEA$_{OCBA}$, the RSA uses fewer samples for PC2, PC6 and PC8, otherwise there is no statistically significant difference. The results also indicate that more samples are required in general for the “harder” problem classes, the even-numbered classes.

In summary, the results indicate that the proposed NEA approaches significantly outperform the baseline approach, the RSA, across all problem classes, but have significantly lower values than those evaluated by the UB procedure. Also, the NEA$_{OCBA}$ approach requires more computational effort than the competing NEA approach, suggesting that the application of the OCBA procedure requires significant additional computation over the sample averaging method of the NEA$_{MEAN}$. The implications of these results are discussed in detail in Section 7.4.2.

7.2.3.3 Validation of Results

A further set of experiments was carried out to validate the results given the approach of evaluating each algorithm once per problem instance. Each algorithm was once more run on each of the problem instances, as before, but using different initial random seeds. This additional set of experiments was intended to provide some assurance that a single algorithm run on each instance was sufficient to provide a reasonable estimate of the performance on the sampled instances of a problem class. The results of these experiments are shown in Figures A.6 and A.7 for solution quality, Figure A.8 for computation time, and Figure A.9 for total number of samples, in Section A.2.1 of Appendix A.

For analysis, nonparametric statistical methods were used to evaluate whether the original and validation results differ significantly. The two-sided Wilcoxon Rank Sum Test for paired samples (Hollander and Wolfe, 1999) was used with the null hypothesis that there are no differences between original experiment results and validation results for each algorithm, each performance criterion, and each problem class. No statistically significant differences were found between each algorithm for each problem class and for each performance criterion at the 0.05 significance level. Overall, this suggests that the use of a single algorithm run per instance (for at least 30 instances) is sufficient to provide a reasonable estimate of the performance of an algorithm for each problem class.

7.3 Opasanon's Safest Escape Algorithm

In this section, the NEA$_{OCBA}$ is evaluated against an existing exact and efficient approach, Opasanon's Safest Escape Algorithm (SEA) Opasanon (2004); Opasanon and Miller-Hooks (2008). The SEA was originally developed to solve a related problem, the Safest Escape Problem (SE), that has the goal of finding a priori flow patterns that maximise the minimum path probability of successful traversal from source to sink, in time-dependent networks with time-varying travel times and stochastic, time-varying capacities[10]. In comparison the MSE, as a reminder, has the goal of finding the flow pattern with the overall maximum probability of successful traversal from source to sink in STV Networks. Additionally, solutions to the SE are permitted to contain cycles, unlike for the MSE, but waiting at nodes, like for the MSE, is forbidden.

[10]These networks are a subset of the more general STV Networks; that is, networks with stochastic travel times that occur almost surely, i.e. with probability 1.

In order to apply the SEA to the MSE in STV Networks, a procedure is required to transform stochastic travel times to deterministic values. Here we use expectation, so that the travel time distributions, for all edges and all time intervals, are transformed to their expected values:[11]

$$\tau_{ij}(t) = \lfloor \mathbb{E}[\bar{\tau}_{ij}(t)] \rfloor = \left\lfloor \sum_{h=1}^{H} \tau_{ij}^{h}(t) \cdot \rho_{ij}^{d}(t) \right\rfloor, \forall (i,j) \in \mathscr{E}, \forall t \in [T]$$

The procedure for applying the SEA to the MSE is then as follows. An instance of an STV Network is transformed to a network with deterministic travel times using the expectation conversion. The SE is then solved on the transformed network using the SEA, and the resulting solution is evaluated on the original STV Network for the MSE. This approach is called the Heuristic SEA (HSEA).

A solution to the SE on the transformed network is not guaranteed to provide the optimal solution to the MSE in the original STV Network. As previously mentioned, the goal of the SE is to maximise the likelihood of the path with the lowest probability of successful traversal; in contrast, the MSE has a system-wide objective to maximise overall likelihood of successful traversal. The difference is that solutions to the SE may include cases where the overall likelihood is lower because the probability of the path with the lowest likelihood of successful traversal is maximised. Also, solutions to the SE are allowed to contain cycles, which are forbidden for the MSE. Finally, the reduction to a deterministic value, here the expected value, as discussed in Section 4.2, cannot guarantee that the optimal solution found for the reduced problem will be the optimal solution for the general problem in STV Networks.

To evaluate the HSEA, its performance was compared with that of the NEA$_{OCBA}$, looking specifically at solution quality and computation time. To compare the approaches directly the HSEA was executed on the same problem instances as the NEA$_{OCBA}$ was run on, for each problem class. Each SE solution was evaluated for the MSE using 1000 samples, and, as with the other approaches, the HSEA was implemented in Java. The experiments were carried out under identical conditions to those detailed in Section 7.2.

The results for the HSEA and the NEA$_{OCBA}$ are shown in Figures A.10 and A.11 for solution quality, and Figure A.12 for computation time. Non-parametric statistical tests were carried out using the Wilcoxon Rank Sum Test for paired samples Hollander and Wolfe (1999) with the null hypothesis that the results of the HSEA and NEA$_{OCBA}$

[11]Values are rounded using the floor function because travel times are assumed to be integers.

do not differ at the 0.05 significance level. The tests indicate that there are strongly significant differences for both criteria, for all problem classes. Test results are provided in Table A.24 in Section A.2.2 of Appendix A. For solution quality, the NEA$_{OCBA}$ find solutions of significantly higher quality than the HSEA as seen in Figure A.13; whereas for computation time the HSEA has significantly lower results, i.e. it is significantly faster, as shown in Figure A.13.

Overall, the results indicate that using the described heuristic procedure, although significantly faster than the NEA$_{OCBA}$, the SEA is not competitive for solution quality with the NEA$_{OCBA}$ for the MSE.

7.4 Discussion

In the final section of this chapter, a number of issues are discussed that have emerged in the course of the experimentation.

7.4.1 Parameter Tuning

The results from the parameter tuning phase present, to the knowledge of the author, the first reported evaluation of the integrated EA and OCBA framework over an EA run. Due to this, a number of issues relating to the performance of the NEA$_{OCBA}$ with respect to input parameters have emerged.

7.4.1.1 NEA parameters vs. OCBA parameters

The first point of discussion, noted previously, is that the EA parameters appear to have a larger impact on the performance of the algorithm, i.e. the quality of the solutions found, than the OCBA parameters. As previously mentioned, in general this observation agrees with the idea that the role of the OCBA is to mitigate the effects of noisy fitness evaluations on the search process; whereas the EA is used to effectively balance exploration and exploitation. It follows that for a problem with a large search space, such as the MSE, the choice of EA parameters could be considered more important in defining an effective and robust algorithm. Or, in other words, the EA parameters govern the maximum possible quality that can be obtained; that is, even if at the mathematical limit the noise was reduced almost surely, the performance of the EA would still be limited by the extent of the exploration of the search space, e.g. the size of the population and the maximum number of generations. This is of course part of the trade-off between where computational effort should be spent — on evaluating solutions more accurately or exploring the space, as discussed in Section 3.4.1.

A particular difficulty that arises out of the integrated framework is dependencies of OCBA parameters on NEA parameters. The computational complexity of the OCBA procedure is driven in part by EA population and offspring sizes, that is, the number of individuals input to each selection problem, and, of course, their fitness distributions. However, in general, the larger the set of individuals, the more computationally demanding the selection problem. This is a considerable (but known) limitation of the procedure[12], in particular when balanced against the necessity for larger population sizes to improve overall search success. This issue motivated the proposed optimisation to reduce the computational requirements of the OCBA and the heuristic for speeding up the selection of individuals to which to allocate additional samples each generation (detailed in Section 5.4.10). However, overall, such a limitation may restrict the applicability of vanilla OCBA within certain EA contexts, perhaps requiring additional procedures such as the pre-screening of clearly poor quality candidates or the integration of variance-reduction techniques to improve the efficiency of sampling.

7.4.1.2 OCBA as part of the NEA

A further difficulty is that the tuning of parameters for the NEA_{OCBA} requires the consideration of a number of additional factors beyond those considered for its tuning for a single selection problem, the problem in Simulation Optimisation for which in general the procedure was originally designed. Here a series of selection problems must be solved, one per EA generation, and hence the dynamics of an EA's population will affect the requirements of each selection problem. For example, in general the populations of early generations are likely to include many poor quality candidate solutions with high variance; whereas later populations tend to include similar individuals of higher quality and with lower variance. Under changing population dynamics, in order to get optimal, or merely robust, performance from the allocation procedure, different parameter values may be appropriate at different stages of the search process. Given the aforementioned scenario, for example, initially a small indifference-zone may be unnecessary to differentiate between the solutions and hence rank them correctly, and, ultimately, would waste substantial effort on non-promising solutions. On the contrary, in the later stages of an EA's run, where solutions are expected to be (genetically) sim-

[12] Although Bayesian R&S methods, such as the OCBA, are known to scale better with larger number of individuals than competing frequentist approaches (Chen, 1996).

ilar (and have similar fitness values), a relatively large indifference-zone could have the opposite effect of not expending sufficient effort to distinguish between similar individuals, potentially to the detriment of the search result. This suggests that it may be difficult to select a predetermined, static set of parameter values that can effectively accommodate the population dynamics of an EA run.

7.4.1.3 Effect of Target Confidence Level

For the target confidence level, the results show that for the given range of parameters evaluated, no significant difference in solution quality was noted between the two confidence levels, $\alpha^* \in \{0.5, .05\}$, studied. We argue that as the number of individuals input to a ranking problem, and hence comparisons, increases, the effect of the α^* parameter value diminishes. For k comparisons, to achieve at least a target confidence level of $1-\alpha^*$, the mean[13] probability of all comparisons being correct must be $\geq (1-\alpha^*)^{1/k}$, with the probability of each comparison $\geq (1-\alpha^*)$. However, as $k \to \infty$, we have that[14] $\lim_{k\to\infty}(1-\alpha^*)^{1/k} = 1$, $\forall\alpha^* \in [0,1)$. The effort required to achieve such a target level will depend on a number of factors, including the variance and the sizes of the differences between candidate solutions. To demonstrate this effect, say $\alpha^* \in \{0.9, 0.5, 0.05\}$ and $\mu = \lambda = 200$, then for the $(\mu+\lambda)$ EA proposed here, the (maximum) number of comparisons per generation is $\mu^2 = 200^2 = 40000$. Given the possible α^* values, the mean probability per comparison for each target confidence level is approximately: $(1-0.9)^{1/40000} = 0.999942437$, $(1-0.5)^{1/40000} = 0.999982671$, and $(1-0.05)^{1/40000} = 0.999998718$, respectively. The additional sampling required to achieve the target level $1-0.05 = 0.95$ over $1-0.9 = 0.1$, in practice, provides little gain in the probability of each comparison (and hence the ordering) being correct. In conclusion, as the population and offspring sizes increase, the number of comparisons increases as well, and the choice of α^* value has less impact on the probability of individual comparisons being correct, and, ultimately, less impact on the performance of the OCBA.

7.4.1.4 Setting the Initial Sample Size

An additional issue surrounds the setting of the initial sample size when considering the assumptions and requirements of the OCBA. In principle, it is desirable to re-

[13]Due to the multiplicative nature of the calculation of the PGG value, the geometric mean is used instead of the arithmetic mean.

[14]Since $\lim_{k\to\infty} 1/k = 0$, then for any $n \in (0,1]$, we have $\lim_{k\to\infty} n^{1/k} = n^0 = 1$.

duce the sample size to minimise costs. However, in practice, a certain sample size is necessary to get a good budget allocation, and hence there is a balance to be struck between over-sampling and achieving reasonable estimates for efficient allocation. Furthermore, the underlying assumptions of the normality of fitness distribution places additional requirements on sample sizes, in particular when appealing to the central limit theorem[15]. This is particularly relevant here given the inherent skew in the fitness distributions. Overall, it is not clear how close to normality the fitness distributions should be in order to still be able to harness the efficiency of the allocation procedure.

7.4.1.5 Summary

In summary, the additional parameters of the OCBA procedure add complexity to the parameter tuning phase, not just due to the number of parameters but because of their interdependencies and, so far at least, the lack of guidance in the literature for setting their values. Further study of the interactions between EA and OCBA parameters is required, in particular in consideration of EA population dynamics, and furthermore, an evaluation of the impact of approximately normally distributed fitness distributions on the efficacy of the OCBA, would help resolve some of these issues.

7.4.2 Evaluation

In this subsection, a number of issues are discussed that arise from the evaluation phase of the experimentation. The main points for discussion are the results of the NEA variants, and in general the applicability of the EA and OCBA framework to the MSE.

In this discussion, as in the literature review, the distribution of the sampling error induced through the MC method is termed "noise". This is reasonable since the commonly assumed additive and Gaussian distributed noise with mean 0 and variance σ^2, $\mathcal{N}(0,\sigma^2)$, is asymptotically equivalent.

As a reminder, note that for the MSE, the noise distribution, e.g. the level and distribution of the noise distribution across the space, is unknown a priori. In the literature, in particular for works applying EAs in noisy and stochastic environments, approaches to problems in such environments are often evaluated using artificially introduced noise, thus assumed known and commonly uniform across the space, on

[15]The central limit theorem guarantees asymptotic convergence in distribution of the sample mean to normality assuming i.i.d. samples; however, in practice, the goal is to find high-quality solutions at low cost, that is, using low number of samples and computation time.

deterministic problems, e.g. ONE-MAX or SPHERE problems, with known optimal solutions (Bianchi et al., 2006b). In practice, for noise induced through MC sampling, it is unlikely that the estimated level of noise will be (exactly) uniform across the space. Perhaps a more pertinent question is: to what extent does it vary across the space? And, specifically, whether the variation will mislead, for example, a simple fixed resampling rate approach, in terms of both the ranking of candidate solutions and in the search itself, and thus warrant more computationally demanding approaches such as the OCBA.

7.4.2.1 NEA Variants: *OCBA* and *MEAN*

The results for the NEA variants show that there are no significant differences between the NEA_{OCBA} and the NEA_{MEAN} for solution quality, except for the "easiest" class, PC1. However, they do indicate significant differences for computational time and number of fitness samples; for both criteria, across all problem classes, the NEA_{OCBA} required more. For the MSE, specifically for the problem classes investigated, the results thus suggest that the application of the OCBA procedure adds little in terms of solution quality to the fitness averaging approach, and requires significantly more effort. The NEA_{OCBA} is expected to require significantly more computation than the resampling approach, since the NEA_{MEAN} uses only the initial sample size, $\bar{n}_0$, to estimate fitness values. However, as a result of this additional effort, the NEA_{OCBA} would be expected to produce an increase in solution quality.

One possible explanation for the result is that the chosen sample size is sufficient to rank[16] the candidate solutions correctly, because the variances of the fitness distributions are low and/or the differences large. This would imply that additional sampling, e.g. through the use of the OCBA, would be unnecessary; however, the fact that the OCBA requires substantially more fitness samples for all problem classes, and hence more computation time, suggests that this is not the case. On the contrary, it suggests that in order to achieve the target levels of confidence in the ordering of the comparisons (and hence individuals), additional sampling is necessary. Nevertheless, given that exact fitness values are unknown, this is difficult to establish definitively.

A more likely explanation is that the ranking of individuals according to the required order information for the EA, while not correct, is adequate for the NEA to

[16]Here *rank* is taken to mean to order candidate solutions given the required order information for the EA. This is not necessarily a full ranking but dependent on the mating and replacement selection schemes.

successfully exploit. Certainly, the results suggest that in general the additional computation utilised by the integrated OCBA procedure provides no significant gain in solution quality, even given the order guarantees that it offers. To try to understand this effect, it is necessary to consider the role ordering plays in an EA.

In a deterministic environment it is often assumed that the ordering of individuals is deterministic and can be computed efficiently, and the selection pressure is controlled through the selection operators. However, as noted by others, such as Miller (1997), noise can reduce selection pressure since selection operators no longer know with certainty the ranking of individuals. The effect here is that strong selection pressure of the $(\mu+\lambda)$ replacement operator is weakened and appears less important for the success of the algorithm. That is, that a lower selective pressure is induced through the noise but that this does not appear to a have significant impact on the performance of the algorithm (as seen by comparison of the results of the NEA_{OCBA} and the NEA_{MEAN}). Assuming this is the case, then the apparent success of the NEAs could be, in part, due to the seeding of the initial population, which works to bias the exploration of the search space with candidates that are at least feasible on a single randomly selected state, and also the final selection procedure that guarantees to some confidence level that the returned solution is the best of the final population.

To investigate the impact of the seeding and final selection, the NEA approaches were once again evaluated, but for these supplementary experiments no seeding of the initial population was carried out, so that the initial population was generated approximately uniformly randomly from the search space (using the methods described in Section 5.1), and no final selection procedure was executed. The only concession was to evaluate the best individual, based on observed fitness values, from the last generation with a further 1000 samples in order to get a more accurate estimate of its true fitness value. The revised NEA_{OCBA} and NEA_{MEAN} are called NEA^{NS}_{OCBA} and NEA^{NS}_{MEAN}, respectively. The results from the additional experiments are provided in Figures A.14 and A.15 for solution quality, Figure A.16 for computation time, and Figure A.17 for total number of samples, in Section A.3.

For analysis, we compared the results of NEA^{NS}_{OCBA} with NEA^{NS}_{MEAN}, and also NEA^{NS}_{OCBA} with NEA_{OCBA}, and NEA^{NS}_{MEAN} with NEA_{MEAN}. For the algorithms not utilising seeding or a final selection procedure, a two-sided Wilcoxon Rank Sum Test for independent samples was executed for each problem class and performance criterion. At the 0.05 significance level, the tests showed significant differences for a number of problem classes for solution quality, and significant differences for all classes for computation time and number of fitness samples. Specifically, the results showed that the NEA_{OCBA}^{NS} performed better than the NEA_{MEAN}^{NS} for the "harder" classes, that is, PC2, PC4, PC6 and PC8, and required significantly more computation time and fitness samples for all problem classes. The p-values for the tests are provided in Table A.25 of Appendix A. These results suggest that the additional sampling is beneficial for the "harder" problems, in terms of solution quality, but it comes at a significant cost. Note that the NEA_{MEAN}^{NS} performed particularly poorly on the PC6 and PC8 where it did no better than the RSA.

The results for each algorithm were then compared with the results of the NEA_{OCBA} and NEA_{MEAN}, respectively, using a two-sided Wilcoxon Rank Sum Test for paired samples, with the null hypothesis that there are no differences between the algorithms for all performance criteria and problem classes. The p-values are provided in Tables A.26 and A.27, in Section A.3.

For the NEA_{OCBA}, the original algorithm performed significantly better for solution quality than the NEA_{OCBA}^{NS} for all problem classes, except PC6 and PC8. For computation time, there were significant differences for the same classes, where the NEA_{OCBA}^{NS} required significantly more computational effort. Finally, for the number of fitness samples, NEA_{OCBA}^{NS} required significantly more fitness samples for classes: PC2, PC4, PC5, PC7 and PC8. Boxplots of the differences can be seen in Figure A.18 in Section A.3.

The results for the NEA_{MEAN}^{NS} compared with the original NEA_{MEAN} indicate that the original algorithm performed significantly better for all problem classes. For computation time, the original algorithm required more computation for PC2, PC3, PC4, PC6 and PC8. Then, finally, for number of fitness samples, with the only difference between the variants utilising samples being the final selection procedure, the original algorithm, the NEA_{MEAN}, used significantly more samples for PC1, PC2, PC4, PC6 and PC8. Boxplots of the differences can be seen in Figures A.19 in Section A.3.

Overall, the results demonstrate that the use of the additional features, namely the seeding of initial population and the final selection procedure, are beneficial to the performance of the NEA, improving the solution quality and reducing the required computational effort. Furthermore, in terms of the gain of using the OCBA over the MEAN approach for the revised algorithms, we would conjecture that the seeding of the initial population reduces the required selection pressure, altering the balance of the conflict between exploitation and exploration, through its biasing of the starting point for the search; ultimately reducing the advantages of using the OCBA observed for the results of the algorithms not using seeding and final selection. The OCBA guarantees at some level the confidence in the ordering of the individuals but at an additional cost, whether it is required or not for successful search.

7.4.3 Random Search Algorithm

As discussed in Section 4.4, a lower bound for optimal solutions can be established by any p-feasible candidate solution. Here a lower bound is established by the RSA (defined in Section 6.2) that acts as a baseline performance for the NEA-based approaches. The RSA is designed identically to the NEA$_{OCBA}$ approach, including seeding of the population, the evaluation of current and offspring populations using the OCBA procedure, and also the use of the final selection method. The only difference between the algorithms is that the genetic operators are essentially disabled for the RSA so that the algorithm performs no hill-climbing, or, in other words, does not exploit knowledge gained during the search. No crossover is performed to generate offspring from the current population, and mutation is executed with probability of 1, that is, all genes are mutated almost surely, so as to generate new approximately uniform random candidates from the search space. The approach was defined in this way to limit the differences between the algorithms and thus allow for a fair comparison with the NEA-based approaches.

The empirical results indicate that both NEA approaches perform significantly better than the RSA in terms of solution quality. This suggests that the NEA genetic operators perform well for the MSE. In terms of computation time, the RSA required significantly more than the other approaches, except for PC6 where it required less than the NEA_{OCBA}, suggesting that in general worse solutions require more computational effort to evaluate and rank. In terms of the number of fitness samples, the RSA required more than the NEA$_{MEAN}$ for all problem classes, but this is to be expected, since it uses the OCBA procedure to rank candidate solutions, and it used significantly fewer than the NEA$_{OCBA}$ approach for PC2, PC6 and PC8, and with no significant difference for the other classes.

7.4.4 Stochastic Upper Bound

In order to gauge the performance of the solution approaches, a stochastic upper bounding procedure, the UB, was proposed. However, as previously discussed, it is difficult to establish the effectiveness of the UB procedure as an upper bound for optimal solutions. Specifically, for the work described herein, it is difficult to quantify the success of the NEA approaches when compared with the UB results, since it is unknown how optimistic a bound it is. In fact, even if the NEA results are optimal, there may still be significant differences with the UB results. Nevertheless, the UB results do indicate, as with the results of the other algorithms, that the odd-numbered problem classes, that is, those with more available supply, are “easier” than the even-numbered ones, the ones with lower supply.

In terms of the effectiveness of the UB itself in determining the feasibility or infeasibility of states, the results show that the heuristics determined that 2989 out of 48000 states (or 6.23% of all states) were infeasible; whereas only 65 (or 0.14% of all states) were discarded due to indecision about feasibility. The success of the heuristics in determining infeasible states implies that the majority of bottlenecks for the studied problem instances occur at the source node.

Future work could include the implementation of a more useful bound, that is, a less optimistic bound. An extension of the bound has been suggested in the literature; however, it relies on being able to efficiently solve subsets of deterministic problems to optimality rather than individual states. As suggested by (Gutjahr et al., 2000), a metaheuristic, such as an EA, could be used to solve such problems, though not necessarily to optimality of course. Nevertheless, this could in theory still provide a less optimistic and perhaps more useful bound on optimal solutions for the MSE. The improved bound applied to the MSE has not been implemented but is formally sketched in Section 8.4.2.2.

7.5 Summary

This chapter has detailed the empirical evaluation of the proposed approaches for solving the MSE. It began with important details about the implementation, including the problem instance generator and, specifically, how problem instances are generated. Then the rest of the chapter details the empirical investigation carried out to evaluate the proposed solution approaches.

The experimental design is detailed, including the problem classes of interest, the performance measures and the general experimental procedure. The problem classes are chosen to represent a variety of scenarios, and solution quality, computation time and the total number of fitness samples are selected as the performance measures of interest.

The first phase of experimentation involves a parameter tuning of the main approach in order to find sets of parameters that lead to robust algorithm performance. This step of the experimentation was also utilised to study the main approach under different parameterisations.

Once parameter tuning was finished, the second empirical phase was carried out with the goal of comparing the NEA approaches against a baseline established by the random search algorithm. The results demonstrated that the NEA approaches, the NEA_{OCBA} and the NEA_{MEAN}, can be used to find solutions that are significantly better both in solution quality and in computational effort than the baseline established by the RSA. Between the two NEA approaches, no significant difference was seen; however, further experiments investigated the circumstances, in particular the seeding of the initial population and the final selection, under which, in general, the NEA_{OCBA} did perform better than the NEA_{MEAN}. The approaches were also compared against the stochastic upper bound, and the results showed that the approaches found results closer to the bound for the “easier” classes, but that it was difficult overall to establish how optimistic the bounds are. Nevertheless, the bound proved useful for indicating the comparative difficultly problem classes.

The main approach was also evaluated against an existing, efficient algorithm originally developed to solve a related problem. The efficient approach was wrapped within a heuristic procedure allowing the approach to be applied to the MSE. Experimental results indicate that the heuristic procedure is significantly faster than the NEA_{OCBA} but finds significantly worse results.

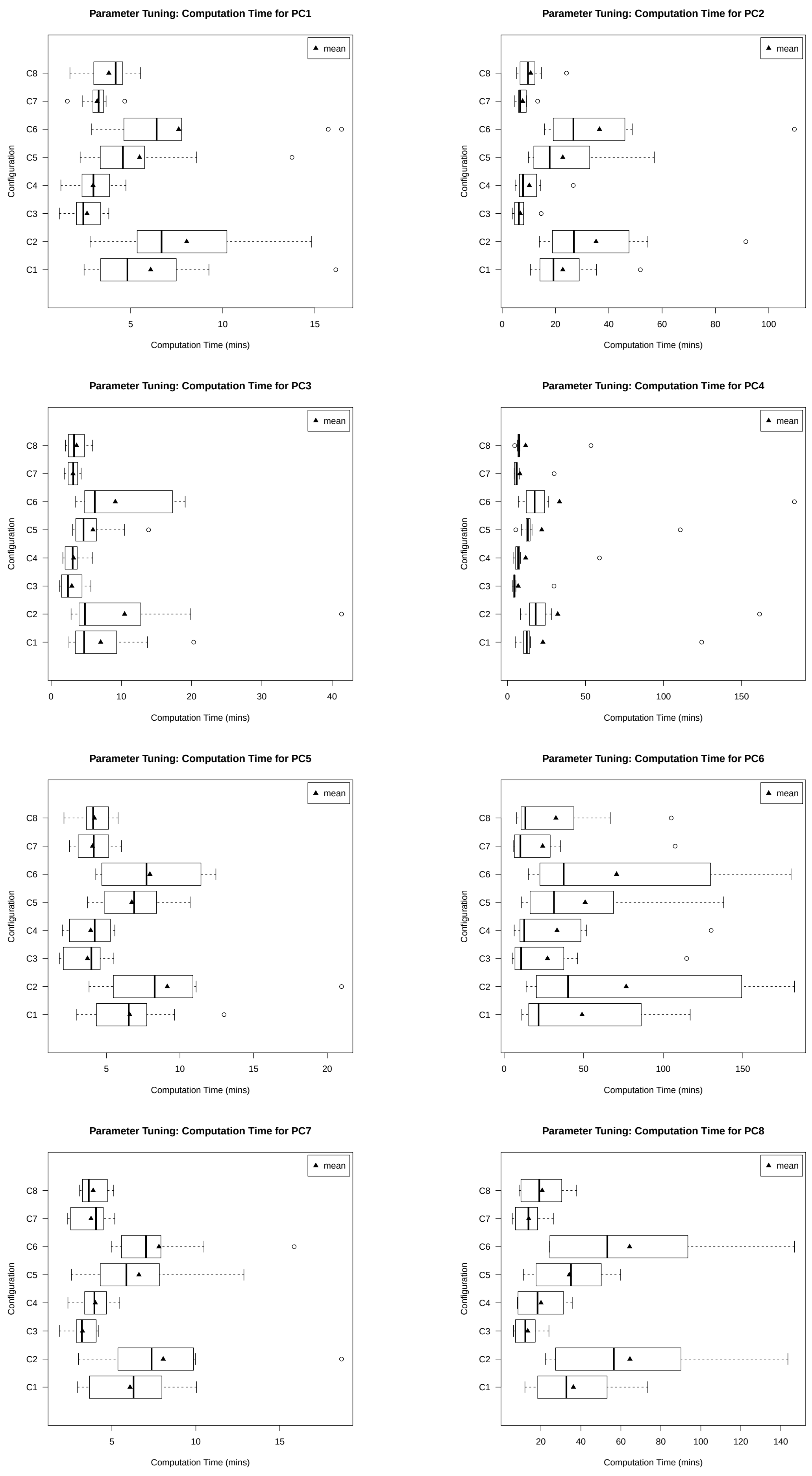


Figure 7.3: Parameter Tuning: CT for PC1-PC8, C1-C8

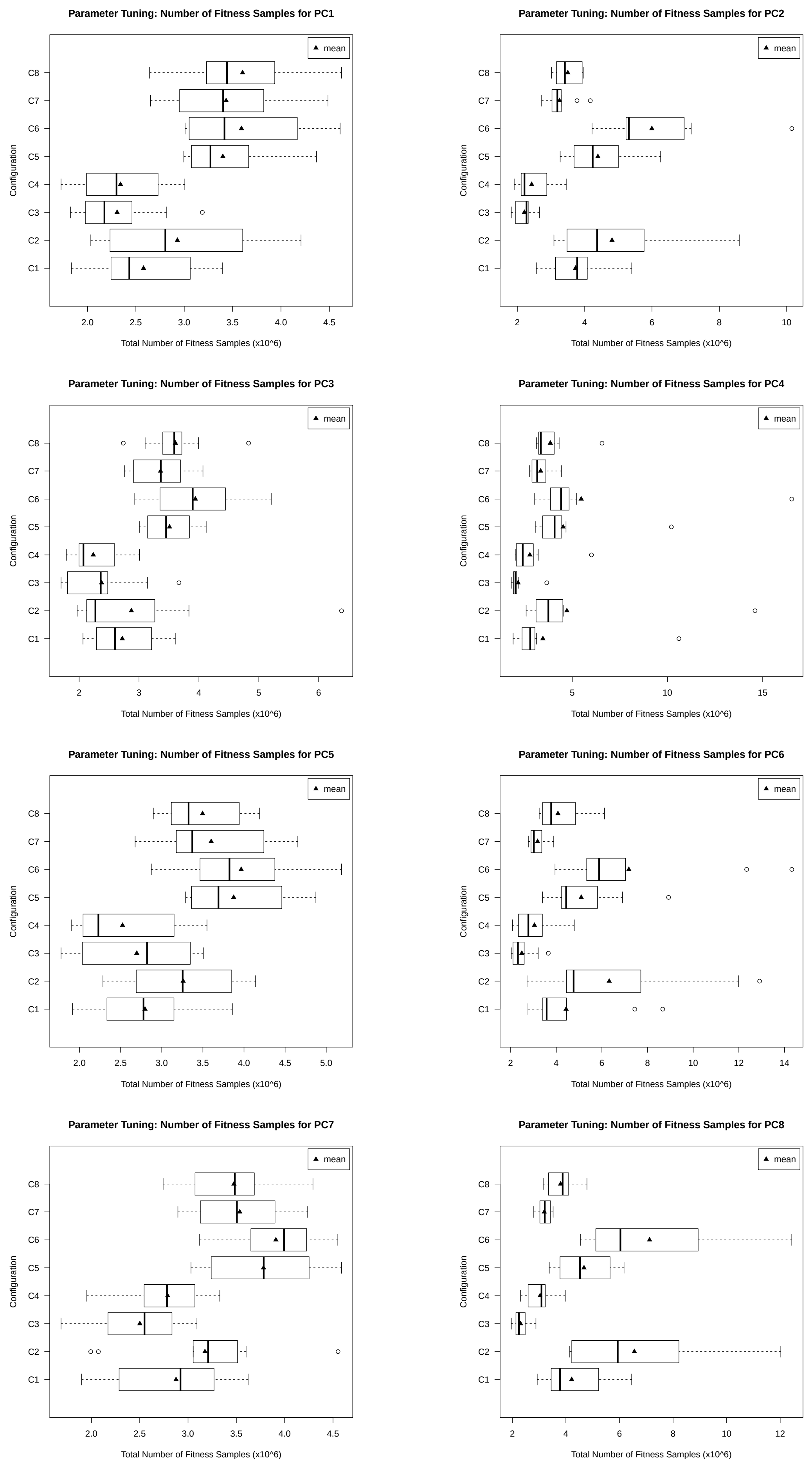


Figure 7.4: Parameter Tuning: N for PC1-PC8, C1-C8

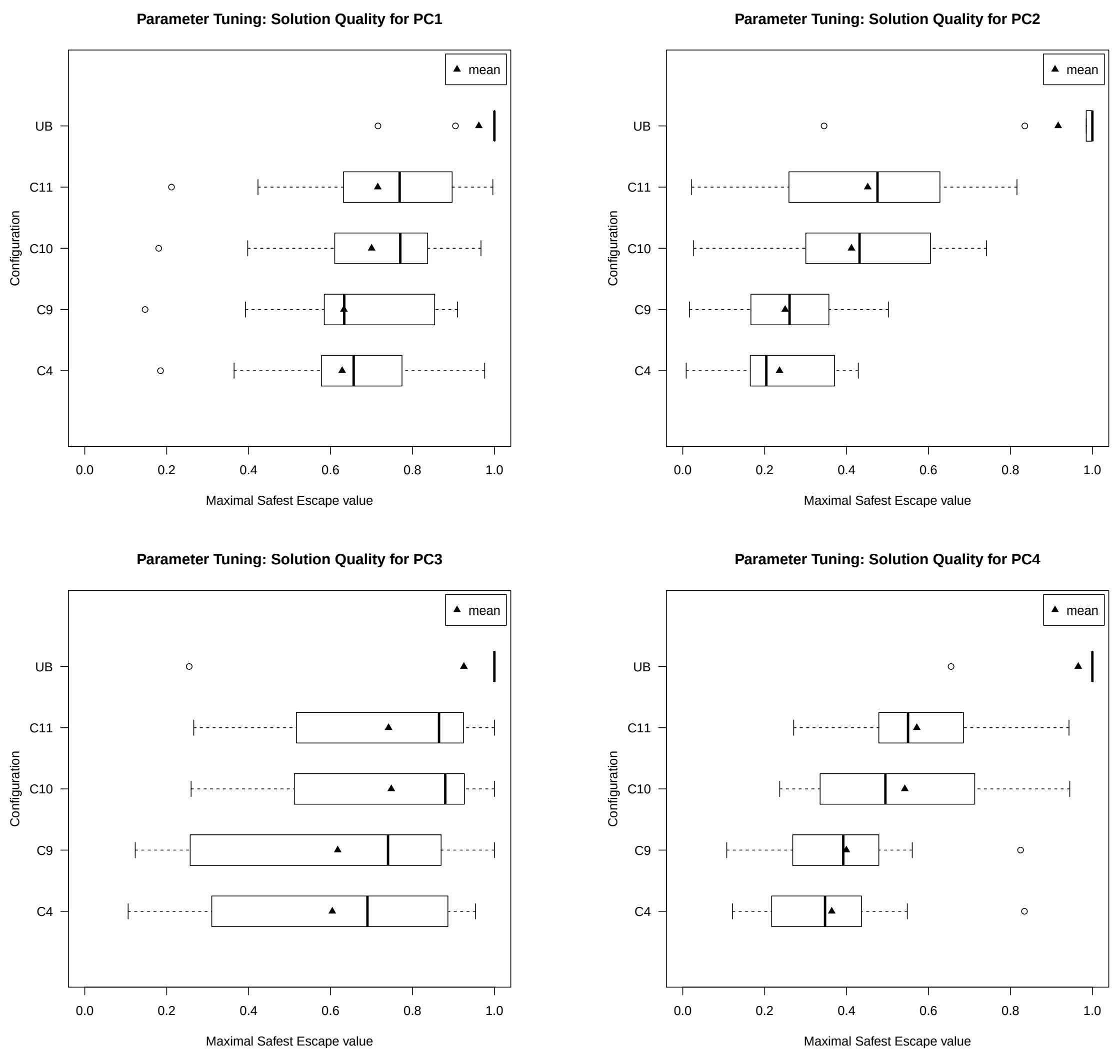


Figure 7.5: Parameter Tuning: SQ for PC1-PC4, C4, C9-C11,UB

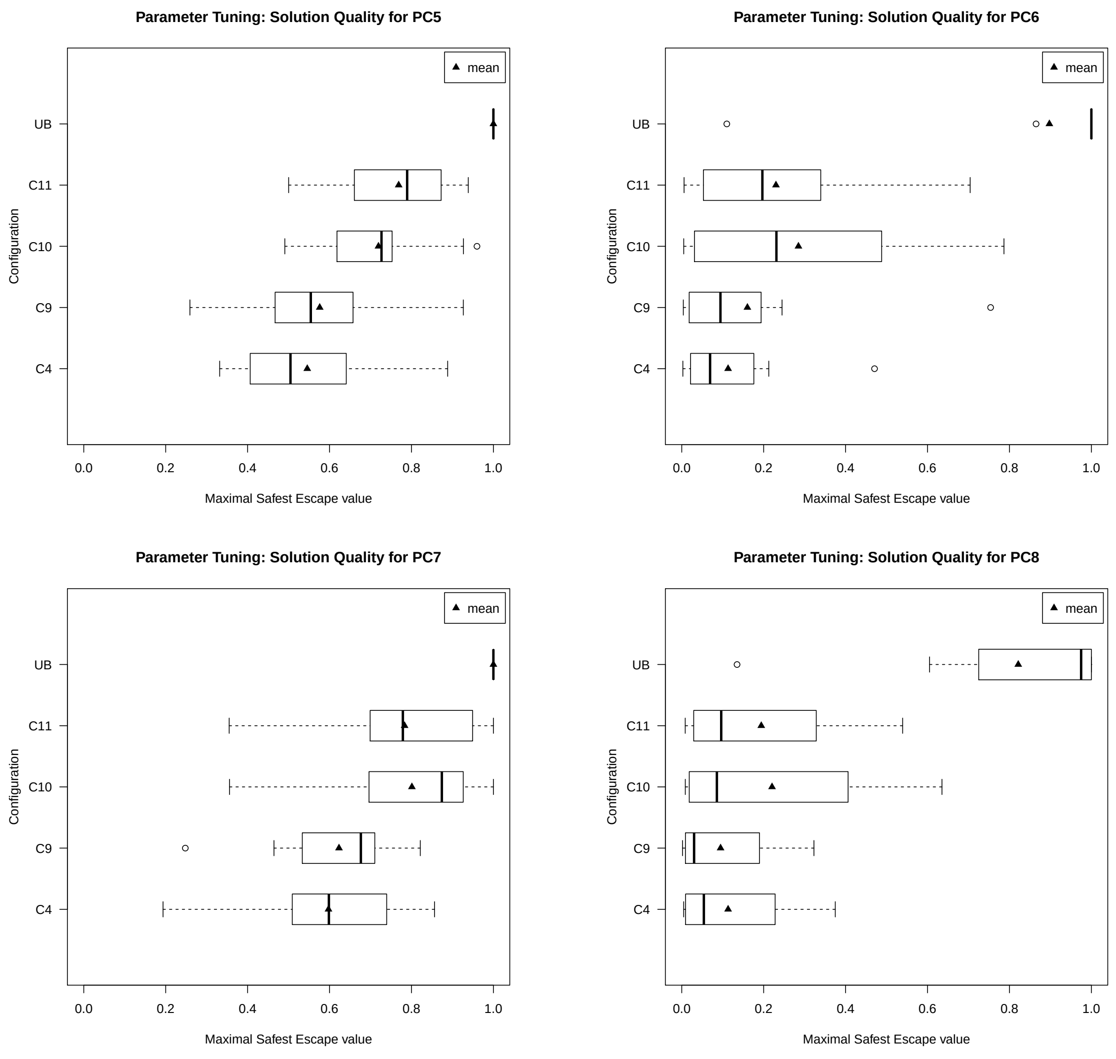


Figure 7.6: Parameter Tuning: SQ for PC5-PC8, C4, C9-C11,UB

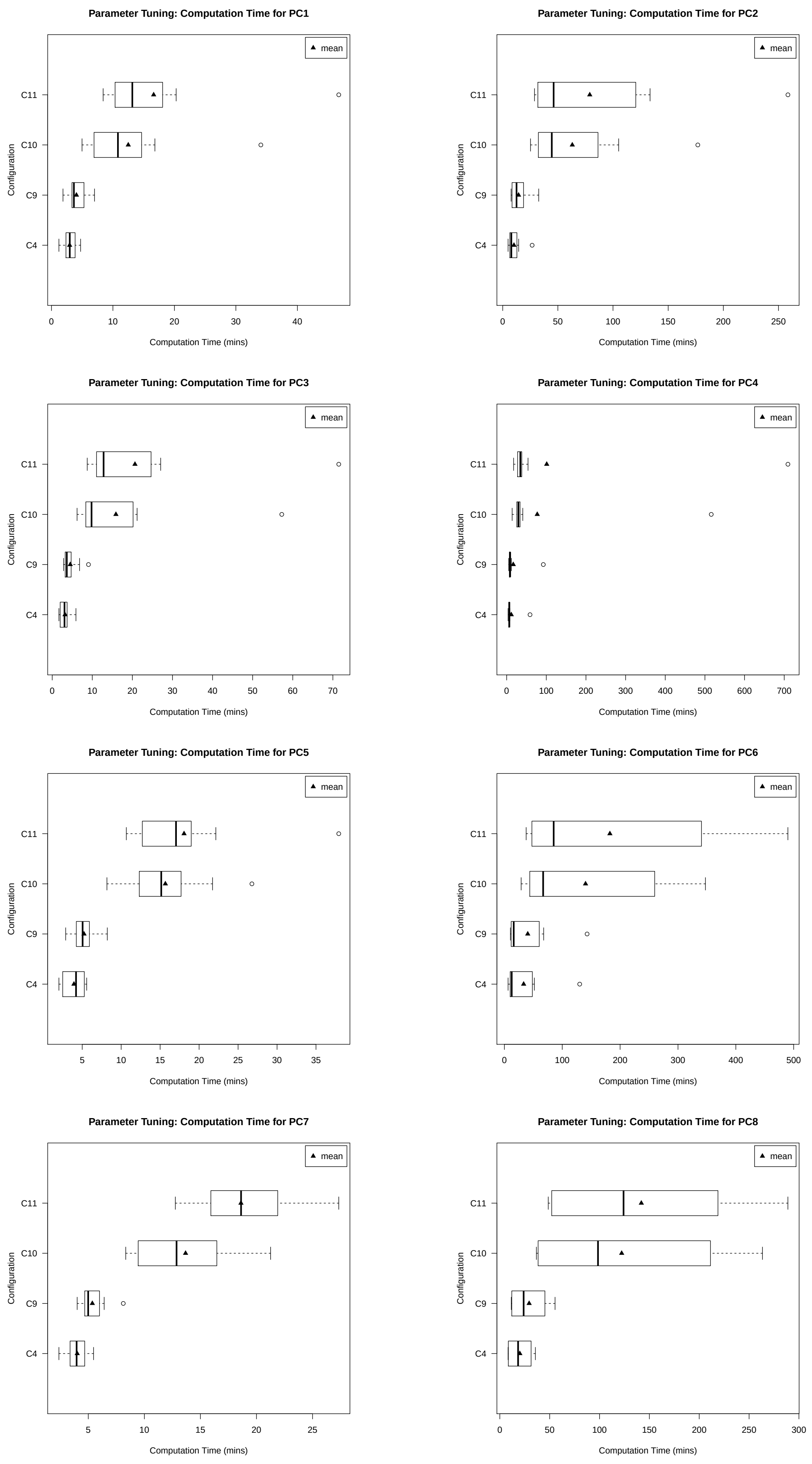


Figure 7.7: Parameter Tuning: CT for PC1-PC8, C4, C9-C11

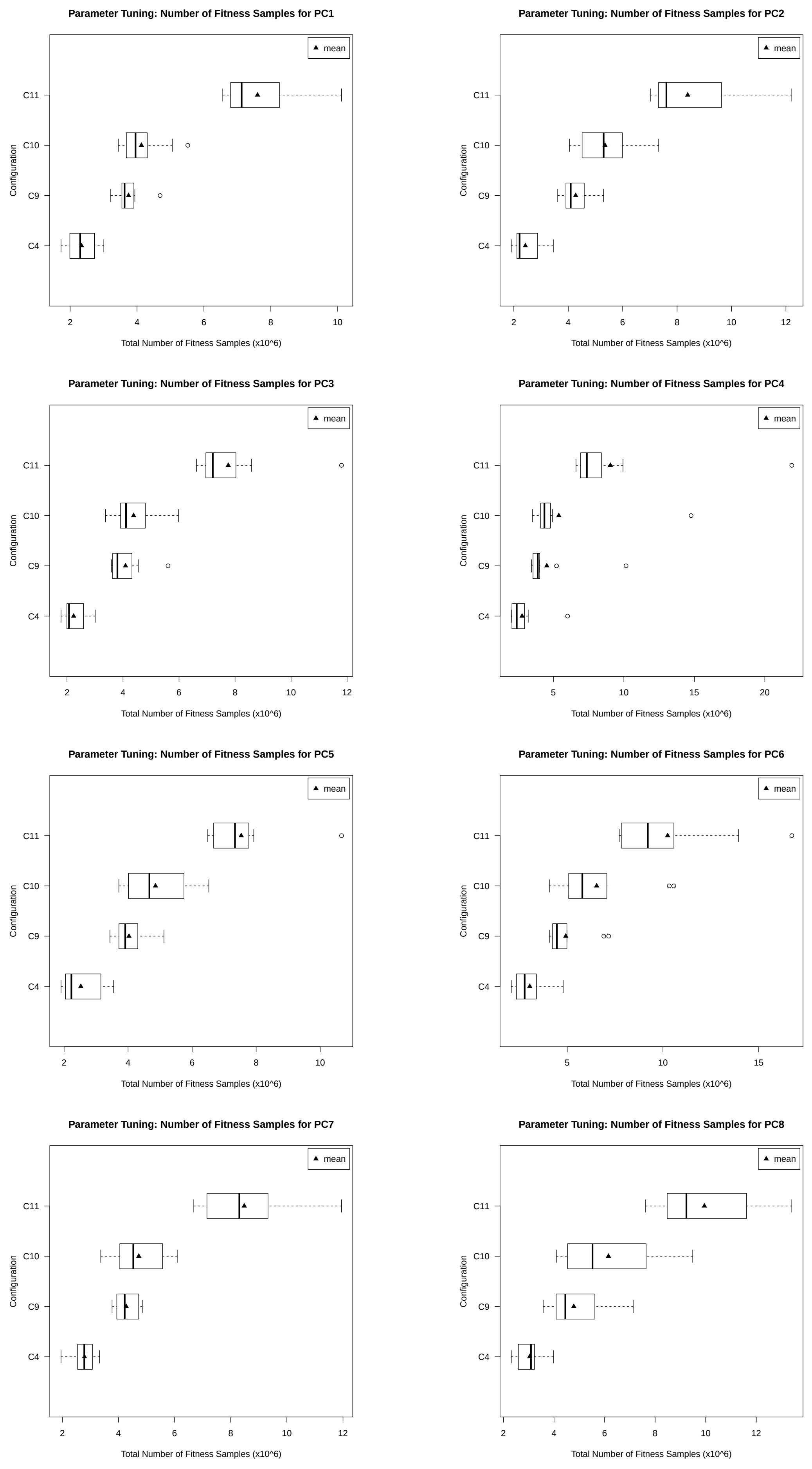


Figure 7.8: Parameter Tuning: N for PC1-PC8, C4, C9-C11

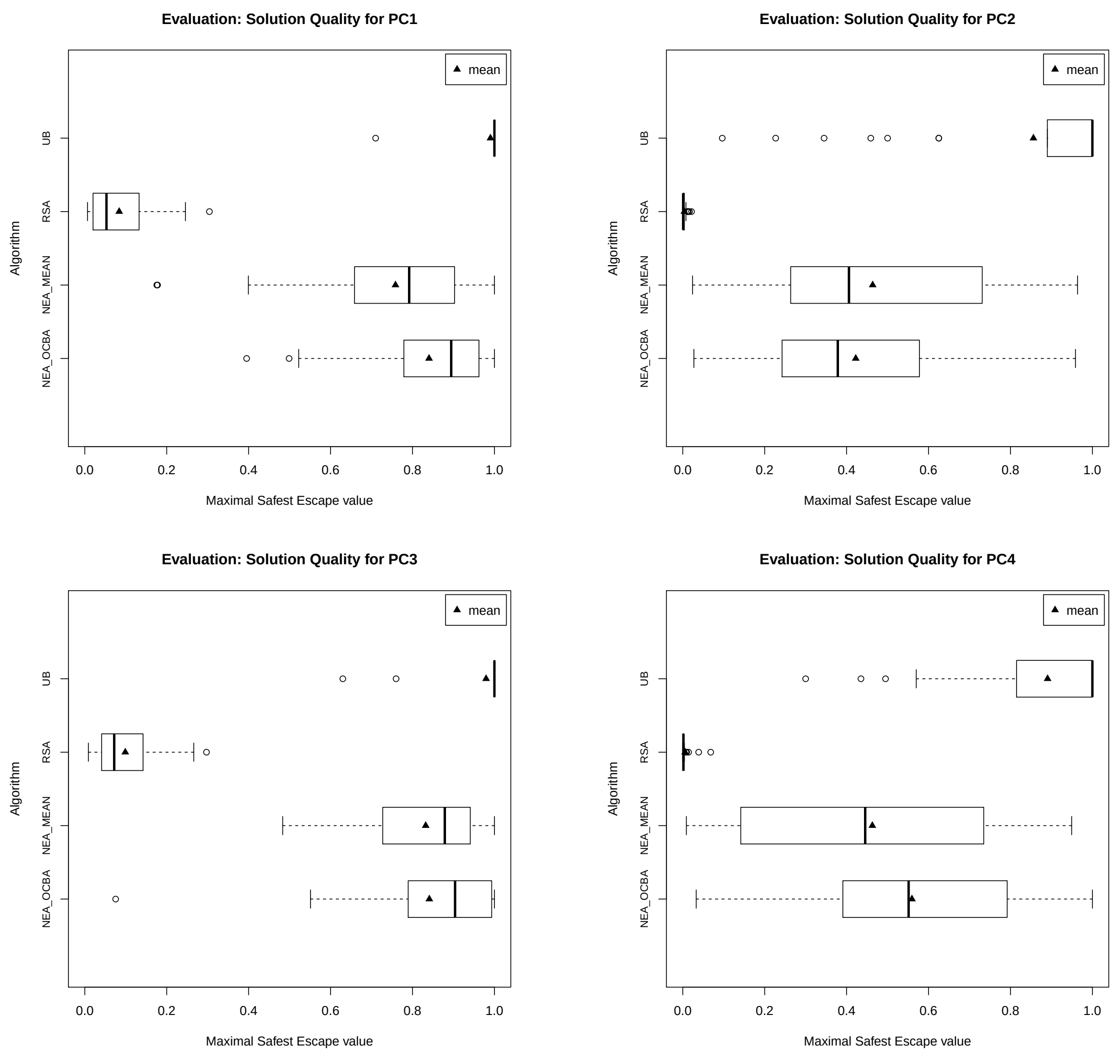


Figure 7.9: Evaluation: SQ for PC1-PC4

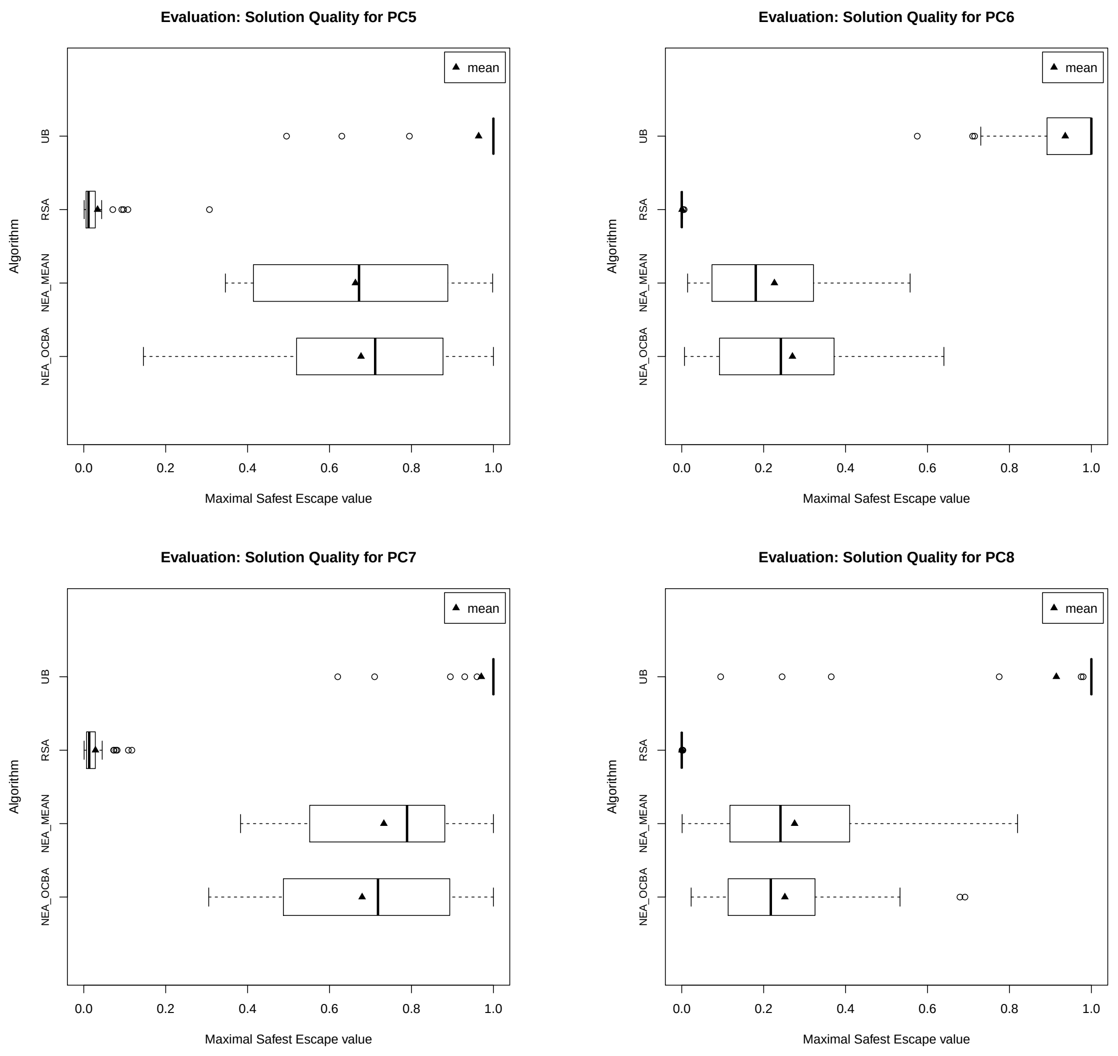


Figure 7.10: Evaluation: SQ for PC5-PC8

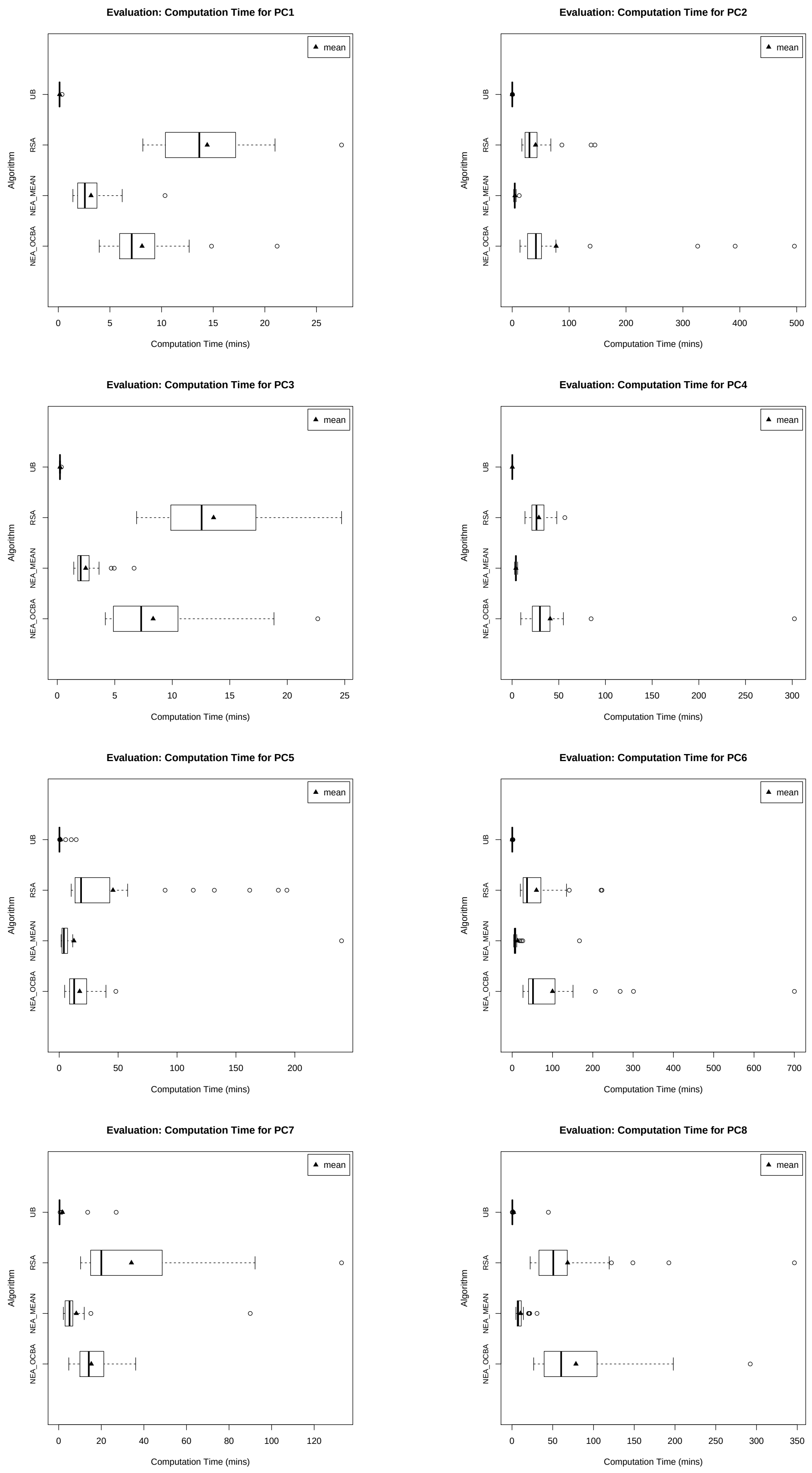


Figure 7.11: Evaluation: CT for PC1-PC8

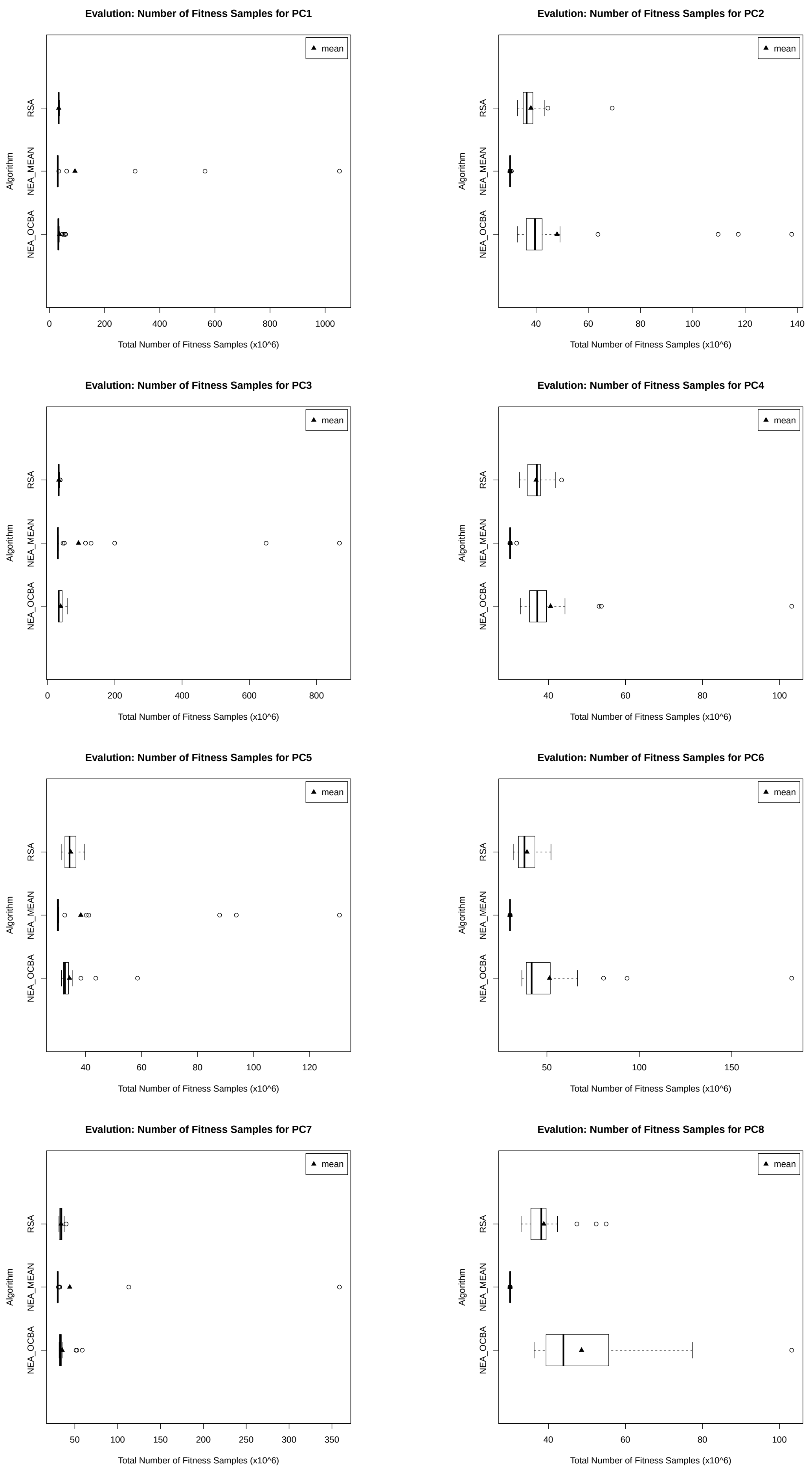


Figure 7.12: Evaluation: N for PC1-PC8

Chapter 8

Conclusions

In the final chapter of the thesis, conclusions are drawn about the work, a summary of the thesis is provided, and the main contributions of the work are discussed in relation to the research hypothesis. This is followed by the identification of potential directions for future research, and the chapter closes with some concluding remarks.

8.1 Summary

This thesis has described an investigation into methods for finding evacuation plans in transient and uncertain environments, in particular the built environment under emergency conditions. It has explored state-of-the-art methods for evacuation planning, focussing specifically on network optimisation approaches. Building on these approaches and in the context of future emergency-response systems, a novel optimisation problem, the Maximal Safest Escape Problem (MSE), has been defined for time-dependent flow networks with stochastic and time-varying edge properties. The definition of the MSE is an attempt to capture the transient and uncertain environmental conditions of, for example, an emergency evacuation scenario in a building.

The MSE and its deterministic counterpart are shown to be NP-hard, and due to this complexity, a general efficient and exact solution approach is unlikely to be proposed. Moreover, evaluating the solution quality of candidate solutions is complex, requiring calculations over potentially high-dimensional spaces. A method for approximating solution quality is developed based on simulation approximation using Monte Carlo sampling methods. For solving the problem, a novel approach, called the NEA$_{OCBA}$ was applied to the MSE. It is based on a state-of-the-art framework integrating Evolutionary Algorithms and a statistical Ranking and Selection procedure, called the Optimal Computing Budget Allocation. For the framework, an optimisation and a heuristic were proposed to improve the computational efficiency of the ranking procedure. A stochastic upper-bounding procedure was also described for providing (optimistic) upper bounds to the quality of optimal solutions.

For evaluation of the NEA$_{OCBA}$, a simpler EA-based approach, called the NEA$_{MEAN}$, using a fitness-averaging method for evaluating candidate solutions, was also defined. Furthermore, to provide a baseline for the EA-based approaches, a straightforward random search algorithm, based on the NEA$_{OCBA}$ but that uses no hill-climbing, was also developed. The main approach was then empirically evaluated against the random search algorithm, the NEA$_{MEAN}$, and upper bounds on optimal solutions. Overall, the results showed that the EA-based approaches perform significantly better than the baseline in terms of solution quality and computation time. Furthermore, in general, the results indicate that the NEA$_{OCBA}$ did not perform significantly better than the simpler NEA$_{MEAN}$; however, conditions under which the integrated framework did perform better than the NEA$_{MEAN}$ were also explored. In a qualitative sense, the results showed that the proposed EA approaches' values were closer to the upper bound values on "easier" problem classes than "harder" ones; the upper bound provided an indication of the comparative difficulty of problem classes. The NEA$_{OCBA}$ was also compared against an existing, efficient approach originally developed for solving a related problem. The existing approach was integrated into a heuristic procedure so that it could be applied to the MSE. Empirical results indicate that although the heuristic procedure requires significantly less computation time than the NEA$_{OCBA}$, it finds significantly worse solutions.

8.2 Main Contributions

This section discusses the main contributions of the thesis in relation to the research hypothesis stated in Section 1.2.1. The order is chronological and the most important contribution is the EA detailed in fourth point.

- Definition of a novel flow problem, the Maximal Safest Escape Problem (MSE), in time-dependent flow networks with stochastic, time-varying edge capacities and travel times.

  The contributions of the thesis begin with the formal definition of a novel flow problem in STV Networks. In the context of future emergency-response systems, the goal of this problem formulation is to capture the transient and uncertain environmental conditions of the built environment during, for example, a hazardous fire scenario, focussing on the evacuation aspect of the response. The idea behind this is to look beyond the use of time as the decisive parameter in evacuation, and adopt a more direct modelling of the risk to occupants during egress.

- Proved that the MSE and its deterministic counterpart are NP-hard.

  The complexity results are an important step in understanding the problem and working towards providing an effective solution approach. The results indicate that it is unlikely (that is, unless P=NP) that efficient and exact algorithms will be proposed for solving the MSE, and consequently motivates the use of approximation techniques, such as metaheuristics.

- Extension of a Statistical Ranking and Selection procedure, the Optimal Computing Budget Allocation (OCBA) method, as part of a state-of-the-art framework integrating EAs and the OCBA applied here to the MSE, with:

  - an optimisation to improve the computational efficiency;
  - heuristic to further improve the efficiency;
  - use of informative Bayesian priors.

  These extensions contribute to improving computational efficiency of the procedure; efficiency is particularly relevant in the context of EAs, given the potentially large number of ranking problems per EA run.

- Design and implementation of an Evolutionary Algorithm for solving the MSE.

  This is the major contribution of the thesis, in particular it provides the first application (and evaluation) of the integrated EA and OCBA framework to a hard optimisation problem, the MSE. The approach may be applicable to similar hard flow problems in STV Networks.

- Empirical evaluation of the application of the EA to the MSE against a competing EA, a random search algorithm, a stochastic upper-bound, and an existing approach developed for solving a related problem.

This is an integral part of the contribution of the thesis and evaluation of the hypothesis. The empirical evaluation of the EAs demonstrates their value and suggests that further investigation could be fruitful.

These contributions were detailed in the previous chapters and contribute towards the evaluation of the research hypothesis. To this end, the results have demonstrated that Evolutionary Algorithms can find *useful* solutions to the MSE, that is, those solutions that are better than the baseline in terms of both solution quality and computational effort.

Within the investigation of the hypothesis, two other research questions were also addressed: one, looking at the gain of using the OCBA over a simple fitness averaging approach; and, two, applying a simple, optimistic upper bounding procedure in order to provide bounds on optimal solutions in order to gauge the performance of the EAs from above as well as from below. The results from these investigations are promising and suggest further research.

8.3 Modelling Assumptions

In this section, we provide a discussion on the assumptions underlying the network model adopted for the work described herein, in the context of their realism and, where possible, mention their impact on the problem and solution approach. These assumptions are defined as part of the adopted network flow model, as formally defined in the Notation and Preliminaries section (4.1).

The following assumptions are discussed:

1. Discrete time horizon.
2. Edge property probability distributions:
    - Discrete values.
    - Known capacity and travel time distributions at time $t = 0$.
    - Statistically independent across space and time.
3. Network supply: fixed and known a priori.
4. Occupant modelling:
    - homogeneous units of flow.

   - no explicit interaction.

5. Solution constraints:
   - No cycles allowed in paths.
   - No waiting at nodes is planned for.

6. Sharing risk amongst occupants.

8.3.1 Discrete time horizon

In practice, on a machine, the discretization of time, as with other variables, is necessary for computation. Hence discrete time, given appropriate intervals, can be used to model continuous time, and in this sense, is a general approach for modelling a time horizon.

8.3.2 Edge property probability distributions

In the network model adopted, network edge property distributions are modelled using discrete distributions, as is common in the network flow area (as discussed in Chapter 2). The choice is based on the simplicity of handling such distributions computationally, in particular when combined with assumptions about the statistical independence of random variables (more on this later).

An important assumption is that the capacity and travel times distributions are assumed known at time $t = 0$, in particular, for future states, i.e. at time $t > 0$. Non-emergency conditions, i.e. normal working conditions, and drills for example, might be used to gauge aggregate values and provide empirical distributions for edge properties. Under normal conditions then, these values could provide reasonable estimates of future distributions (under non-emergency conditions, of course).

For future states, i.e. $t > 0$, during an incident, it is the wider context of future emergency response systems, as envisioned by, for example, the FireGrid project, in which such a system would find itself situated. Within the infrastructure of such systems, it is expected that a multitude of predictive tools would be available to provide forecasts about, say, environmental conditions, for example, the conditions of building circulation systems during fire incidents. These tools could include models that predict smoke layer height and determine the level of toxicity in particular corridors or stairwells. Edge capacities and travel times could then be defined as functions of these conditions. As the conditions change, the edge values would be updated accordingly. In practice then, given updates of edge property distributions and occupants locations, additional optimisations would be required to adjust to the transient conditions.

Another modelling assumption for edge property distributions is that all distributions are defined as statistically independent across both time and space, that is, over all time periods and edges. This is a common assumption for simplifying computations across joint distributions of large numbers of random variables. In practice, however, dependencies across space and time may exist. For example, the travel time required to traverse an edge may be affected by the number of individuals traversing the edge, which in turn is limited by the edge capacity. The sampling approaches proposed herein could, in theory, be adapted to handle such dependencies.

8.3.3 Network supply

The supply of flow to the network is assumed to be fixed and known with certainty a priori, that is, at time $t = 0$. The main assumptions here being, firstly, that the supply is known, and, secondly, that no new flow is introduced or removed during the optimisation. As for the first assumption, again within the context of FireGrid, provision would be made for the tracking of occupants movements, as would clearly be a prerequisite for such an egress operational planning system.

As for the second assumption, the proposed approach carries out an a priori optimisation, that is, at $t = 0$, and hence the flow within the system is assumed fixed, and additional optimisations, $t > 0$, could account for additional flow, and indeed for removed flow (occupants reaching places of safety). However, note that the model allows for the modelling of flow departing at $t > 0$, capturing the idea that individuals may delay immediate egress, due to a slow reaction to an alarm signal (as seen in practice and documented in the egress literature, for example, SFPE (2002)).

An appropriate model extension might explicitly model the uncertainty in the departing times of flow, and model departure times as stochastic; the solution approach would require to be extended to handle this case.

8.3.4 Occupant modelling

The modelling of occupants takes a reductionist approach, whereby individuals are modelled homogeneously, that is, no individuals characteristics are taken into account. This is standard for network flow models applied to evacuation, where it is common for individuals to be modelled as single, independent units of flow with assumed identical characteristics.

An additional simplification in this model is that units are assumed to have no explicit interaction, only an implicit interaction through the shared use of edge capacity. The approach parallels early approaches to modelling and simulation of human behaviour during egress Gwynne et al. (1999).

It would be possible to increase the sophistication of the occupant modelling, for example, introducing distributions for population characteristics, or social aspects, but this would have a significant impact on the solution approach. There is, of course, a trade-off between increasing the complexity of the model and the computational complexity of solving problems.

8.3.5 Solution constraints

A number of constraints are placed on solutions. The first constraint being that solutions are forbidden to contain cycles. From the modelling perspective, this is a reasonable assumption given that individuals are unlikely to be required to trace back over previous steps during an evacuation. Additionally, from the algorithmic perspective allowing cycles increases the search space substantially[1]; however, it potentially allows for the existence of solutions to problems that do not permit solutions without cycles.

Related to cycles is the assumption that waiting at nodes is forbidden[2]. This is restrictive as plans cannot be proposed that allow flow to wait at nodes for a number of time periods, or, in other words, proposed solutions will not plan for occupants to wait at decision points as a solution to, say, congestion on the next edge for traversal. However, again, allowing waiting would increase the search space substantially. Nevertheless, and perhaps more importantly, the inclusion of waiting in STV Networks raises a number of modelling issues. A sample of these issues are:

- Should waiting be allowed at all nodes, or, say, just source nodes or major intersection points?
- Do nodes need capacity values?

[1]Technically, for an infinite time horizon ($T = \infty$), there are an infinite number of paths containing cycles; with finite time horizons ($n < T < \infty$), there are still substantially more paths with cycles than without.

[2]Note that, strictly, waiting is forbidden by the restriction on paths containing cycles, since self-cycles are a subset of general cycles. However, in the context of emergency evacuation, given the importance of waiting, and in (S)TV Networks given additional algorithmic requirements, it is often considered a separate issue.

- Should waiting, as part of the plans, be modelled as time-invariant or time-dependent?
  - Given that paths in the work described herein are time-invariant, should waiting be invariant, such that units of flow wait the same number of time intervals at a node in the path, no matter what time they arrive at the node?
  - Or should waiting be time-adaptive, and hence require paths to be time-adaptive as well?

Given the importance of waiting, the no waiting assumption could be relaxed. However, allowing waiting could require substantial extensions to the model and solution approach, including modification of EA solution representation and genetic operators.

8.3.6 Sharing risk

The final assumption for discussion is that routing plans can be returned that require occupants arriving concurrently at a decision point to be split to continue their egress on differing routes. The main issue here, besides issues of communicating the different routes and social aspects making it unlikely for occupants to split up, is that the risk attached to the paths will most likely differ. In general, this raises a number of issues to do the allocation of risk amongst occupants when there are choices.

A possible extension would be for the model to anticipate that groups will not split, possibly combined with waiting to prevent unrealistic edge capacity requirements. This extension would require substantial changes to the approach to accommodate both the group constraint, and, in particular, waiting, as mentioned previously.

8.4 Future Work

In the course of the work reported here a number of potential areas of further work have been identified, and they can be categorised into the following sub-areas:

- Network Model and Problem
- Algorithms
- Evaluation

8.4.1 Network Model and Problem

In an effort to add realism to the network model, a number of potential extensions are possible. For example, currently the supply of flow at the source node is deterministic; however, similar to network edge properties, the availability of supply could be modelled stochastically, that is, that the availability of supply at the source node for a departure time t, $b_s(t)$, is a priori unknown with certainty.

A further adaptation of the model would be to allow waiting of supply at source and intermediate nodes. The addition of waiting could raise modelling issues, for example, with the use of a priori fixed paths, and hence it may be beneficial to investigate the use of time-adaptive paths (mentioned in Section 2.5.5).

Towards the goal of providing online plans, the approach could be adapted to execute adaptive optimisation using updated distributions from, for example, analytic tools such as the ones mentioned in Section 2.2 of the Literature Review.

Modifications to the network model such as these would have implications for the solution approach, and understanding the impact on the methods would become an element of future work as well.

Another avenue would be to extend the main solution approach and apply it to canonical flow problems, e.g. maximum flow and minimum cost, and multiobjective problems, in STV Networks.

8.4.2 Algorithms

There are a number of potentially interesting directions for improving the proposed solution approaches and the upper bound.

8.4.2.1 Noisy Evolutionary Algorithms

For the integrated framework, future work could aim to make the approach more efficient or, perhaps, increase its applicability. Additional extensions could include adapting the OCBA to work with non-normal distributions. In the context of EAs, perhaps using distributions that model, for example, the skew that is common due to fitness functions with additional penalty terms for constraint handling. Or, perhaps, modifying the approach to function with nonparametric distributions in order to widen its applicability. In terms of efficiency, one improvement would be to allow the use of variance-reduction techniques in order to improve the efficiency of sampling and comparisons. Additional optimisations of the OCBA may also be possible, for example, by re-using more information across stages of the OCBA for a single ranking problem or across multiple ranking problems. Finally, further experimentation of the integrated framework is still warranted, in particular focussing on the interactions between EA and OCBA parameters, and in consideration of EA population dynamics (as discussed in the Discussion section (7.4) of Chapter 7).

The efficiency of the NEA$_{MEAN}$ could itself be improved by, for example, using variance-reduction techniques for fitness sampling or adaptive sampling policies.

Additional experimentation would also be beneficial for both exploring the effectiveness of the NEA$_{OCBA}$ and to gain further insight into the circumstances that warrant the use of sophisticated evaluation methods, e.g. R&S methods, over simpler methods such as fixed resampling.

8.4.2.2 Stochastic Upper Bound

For improving the upper bound, in order to provide a better gauge of the performance of the algorithms, future work could include improving the estimate of the current bound using, for example, variance-reduction techniques such as stratified sampling, or, perhaps, improving the bound itself in order to evaluate a less optimistic bound and hence a better gauge of the performance of proposed approaches.

From (Mak et al., 1999), a common improvement of the bound can be achieved through “inner sampling”, that is, by solving over subsets of the $\bar{\xi}$’s support, rather than individual states. However, this requires optimising over sets of network states, rather than single states, which makes the improvement more computationally demanding. Note that solving over the *entire* support of $\bar{\xi}$ is equivalent to solving the general MSE problem.

Formally, a way to improve the upper bound (4.14) and its MC estimate (4.16) would be to use m independent copies $\bar{\xi}^j$ of $\bar{\xi}$ to obtain

$$\begin{aligned} z^* &= \max_{\psi\in\Psi} \mathbb{E}[h(\psi,\bar{\xi})] \\ &= \max_{\psi\in\Psi} \mathbb{E}\left[\frac{1}{m}\sum_{j=1}^{m} h(\psi,\bar{\xi}^j)\right] \\ &\leq \mathbb{E}\left[\max_{\psi\in\Psi}\left[\frac{1}{m}\sum_{j=1}^{m} h(\psi,\bar{\xi}^j)\right]\right] \end{aligned}$$

Note that if $m = 1$ then this approach is identical to the bounding approach defined in Section 4.4.1. As m increases, the closer the combination of sub-problems comes to the general MSE.

This method leads to the unbiased MC estimator

$$\frac{1}{n}\sum_{i=1}^{n}\max_{\psi\in\Psi}\left[\frac{1}{m}\sum_{j=1}^{m}h(\psi,\xi^{ij})\right]$$

where ξ^{ij} are i.i.d. observations from $\bar{\xi}$, $i = 1,\ldots,n$ and $j = 1,\ldots,m$. In other words, instead of solving Equation 4.5 one maximises the empirical estimates of the expected value function

$$G_m(\psi) = \frac{1}{m}\sum_{j=1}^{m}h(\psi,\xi^{ij})$$

Solving over subsets of states is a more computationally demanding problem, and the upper-bounding approach suggested herein is not appropriate. Gutjahr et al. (2000) suggested using metaheuristic approaches, for example EAs, to solve the sub-problems. However, using EAs would reduce the reliability of the bound since such an approach would not guarantee solving the sub-problems to optimality. Nevertheless, the results could still indicate the general direction of the optimal solution, that is, by indicating how optimistic the original bound is. Variance-reduction techniques could also be applied in order to reduce the variance of the estimates.

8.4.3 Evaluation

As mentioned in the Experimentation and Analysis chapter (7), further experimentation is warranted, and in this section a number of potential areas for further experimentation are highlighted.

As previously mentioned, more experimentation investigating the interactions and dependencies between EA parameters and OCBA parameters would be fruitful. The goal being to provide further insight to the performance of the integrated approach and improving the guidelines on effective parameter choices for the framework. In particular, highlighting how to apply the approach to maximise its effectiveness, and also to identify its limitations.

A further possibility would be to look at evaluating the proposed approaches on additional MSE problem classes. For example, using larger numbers of nodes and more network supply in order to look at how the performance of the algorithm scales with larger problems.

Another interesting, and important direction, would be to consider the domain of interest, and to carry out experimentation on structured problem instances. The work described herein explores the application of methods to worst-case scenarios, when very little can be assumed about, for example, the underlying network structure. It is expected, however, that the algorithms' performance would improve as more domain knowledge is included and can be exploited. This could entail, for example, using underlying network topologies based on prototypical building circulation systems or specific real-world cases.

8.5 Concluding Remarks

This thesis has documented an investigation into methods for finding evacuation plans that address the inherently transient and uncertain environmental conditions of emergency scenarios in the built environment. A novel network flow problem was defined that attempts to model the environmental conditions using stochastic and time-varying network properties. Due to the dual complexity of the problem and of evaluating the quality of solutions, an approximation approach based on Evolutionary Algorithms was proposed. An empirical evaluation of the algorithm against several competing methods demonstrated the value in the proposed approach.

Appendix A

Additional Results

This supplementary chapter provides additional tables and plots of the results from the experimentation, described in Chapter 7.

A.1 Parameter Tuning

These additional tables and plots are discussed in Section 7.2.2.3.

The results for the first eight configurations for solution quality are given in Tables A.1 and A.2; for computation time in Tables A.3 and A.4; and for total number of fitness samples in Tables A.5 and A.6. Boxplots showing the pairwise differences between configurations C1-C8 for computation time and number of fitness samples, for each problem class, are shown in Figures A.1 and A.2, respectively.

The results for the additional three configurations: C9, C10 and C11, and configuration C4, for solution quality are provided in Tables A.7 and A.8, for computation time in Tables A.9 and A.10, and Tables A.11 and A.12 for number of samples. Boxplots showing the pairwise differences between configurations C9-C11 and C4 for solution quality, computation time and number of fitness samples, for each problem class, are shown in Figures A.3, A.4 and A.5, respectively.

A.2 Evaluation

Tables of results for all algorithms for solution quality are provided in Tables A.13 and A.14, for computation time in Tables A.15 and A.16, and, finally, for total number of samples in Tables A.17 and A.18.

Table A.1: Parameter Tuning: Results for SQ, PC1-PC4, C1-C8, UB

	PC1		PC2		PC3		PC4	
	mean	sd	mean	sd	mean	sd	mean	sd
C1	0.5833473	0.2107864	0.2333948	0.1188420	0.6255795	0.3099517	0.3529914	0.2016093
C2	0.5700133	0.2411224	0.2588667	0.1476612	0.6468767	0.3068043	0.3861643	0.1783204
C3	0.6089374	0.2240305	0.2335937	0.1125225	0.657628	0.3101052	0.3722691	0.1969543
C4	0.6283052	0.2138481	0.2362546	0.1236281	0.6042334	0.3054539	0.3636693	0.1970668
C5	0.608481	0.2166964	0.2265229	0.1472895	0.6531124	0.3072102	0.3633775	0.2017356
C6	0.6203887	0.2215766	0.24669	0.1255693	0.6150972	0.3043939	0.4289321	0.2393542
C7	0.5854399	0.1995145	0.2361601	0.1350630	0.6140261	0.2826052	0.4274196	0.2276978
C8	0.6015091	0.2123047	0.2583273	0.1433295	0.6437378	0.3079627	0.3969545	0.1772953
UB	0.9620736	0.08686017	0.9165	0.1966602	0.9255	0.2235	0.9655	0.1035

Table A.2: Parameter Tuning: Results for SQ, PC5-PC8, C1-C8, UB

	PC5		PC6		PC7		PC8	
	mean	sd	mean	sd	mean	sd	mean	sd
C1	0.5561188	0.2318858	0.1268538	0.1510684	0.5931456	0.2220617	0.08237323	0.08321179
C2	0.6126256	0.2198117	0.1517843	0.1830562	0.6498687	0.1958959	0.08315901	0.08949078
C3	0.587705	0.2330571	0.1351565	0.1590093	0.5608708	0.2110650	0.08726909	0.09673084
C4	0.5456021	0.1750339	0.1131496	0.1366600	0.5973881	0.1922473	0.1131510	0.1280046
C5	0.5553821	0.16195	0.1052873	0.1558022	0.5998375	0.2410538	0.0997031	0.1190616
C6	0.6465028	0.1693832	0.1484529	0.1630733	0.624425	0.1893118	0.0883803	0.08993004
C7	0.5746153	0.2005448	0.1309603	0.1631611	0.555947	0.2388323	0.1194001	0.1414964
C8	0.5584347	0.2136062	0.1396286	0.1607955	0.553757	0.1678266	0.09338669	0.1217606
UB	1	0	0.8975	0.2655678	1	0	0.8215	0.2652833

Table A.3: Parameter Tuning: Results for CT (milliseconds), PC1-PC4, C1-C8, UB

	PC1		PC2		PC3		PC4	
	mean	sd	mean	sd	mean	sd	mean	sd
C1	365203.3	234493.5	1365027	731085.2	423293.7	329377.2	1359364	2042367
C2	482430.8	242663.6	2113464	1365828	627724.3	687593	1930477	2607645
C3	157871.8	48563.21	410391.9	176345.6	176669	91919.64	410081.9	460512.8
C4	177465.3	64589.71	612275.6	373842.3	191977.7	80719.48	697703.3	949344
C5	328985.7	192987.4	1364548	856988	357533.7	200871.0	1315690	1783691
C6	456415.1	270855.7	2190828	1601074	548374.1	355400.5	1998816	3028432
C7	190905.2	46649.8	461177.6	138377.4	186925.2	48121.75	475176.2	443649.6
C8	228924.3	71196.69	642053.7	315087.4	217773.7	77101.95	696186.1	839247.3
UB	8750.9	542.1288	9515	832.2423	20876.2	3909.546	19416.9	1095.398

Table A.4: Parameter Tuning: Results for CT (milliseconds), PC5-PC8, C1-C8, UB

	PC5		PC6		PC7		PC8	
	mean	sd	mean	sd	mean	sd	mean	sd
C1	395188.6	177504.4	2939184	2449015	364811.1	137289.5	2173943	1169729
C2	548184	276254.8	4603518	3801040	483418.3	247064.2	3872120	2422912
C3	223151.4	76464.6	1637554	1935137	194891.7	45881.28	803216.3	376170.5
C4	236343.3	80372.76	1995638	2182651	240777.8	52267.79	1202544	640762.7
C5	403600	132530.0	3056332	2456331	396632.2	190816.3	2051718	1089266
C6	477357.2	182375.7	4241388	3383185	468102.5	184823.9	3862290	2503063
C7	243651.7	67090.77	1454268	1771765	225368.2	58879.08	833322	409747.8
C8	251355.9	63167.83	1953066	1842200	233016.4	42745.32	1237474	636032.6
UB	16950.6	1340.55	1201648	2893104	30275	2445.843	47161	53340.03

Table A.5: Parameter Tuning: Results for N, PC1-PC4, C1-C8

	PC1		PC2		PC3		PC4	
	mean	sd	mean	sd	mean	sd	mean	sd
C1	2578936	481162.6	3727984	816055.3	2720410	514776.1	3459558	2408492
C2	2929396	702638.7	4812414	1605014	2872305	1298901	4718371	3358988
C3	2305700	409080.5	2209093	250169.2	2374888	594896.4	2156261	513283.2
C4	2341098	403077.9	2425277	483015.7	2235983	369770.8	2776157	1141218
C5	3399390	410371.7	4393160	884987.2	3509195	374545.9	4522085	1961606
C6	3591555	553660.8	5995770	1624484	3940310	659542.6	5470400	3741930
C7	3432965	544119.1	3252280	406561.2	3359485	428138.1	3340975	526450.5
C8	3603130	585643.8	3498840	374601.2	3607455	526424.1	3849090	989121.8

Table A.6: Parameter Tuning: Results for N, PC5-PC8, C1-C8

	PC5		PC6		PC7		PC8	
	mean	sd	mean	sd	mean	sd	mean	sd
C1	2796022	560299.7	4430238	1872946	2876017	551272.5	4215905	1092509
C2	3260503	616358	6320318	3280744	3175230	704260.4	6555464	2586708
C3	2696505	641168.2	2491348	504261.9	2501418	434162.8	2304872	258548.4
C4	2523214	589835.9	3042367	833657.6	2788673	398244.6	3037142	497190.5
C5	3874450	560814.6	5092505	1585536	3781125	534847.1	4675600	1009386
C6	3965590	687124.7	7179570	3197486	3909030	413733.8	7124095	2564883
C7	3600935	619925	3179915	369245.7	3534815	443431.7	3192770	222707.7
C8	3496195	469716.1	4073570	904323	3475030	473084.9	3802775	468571.6

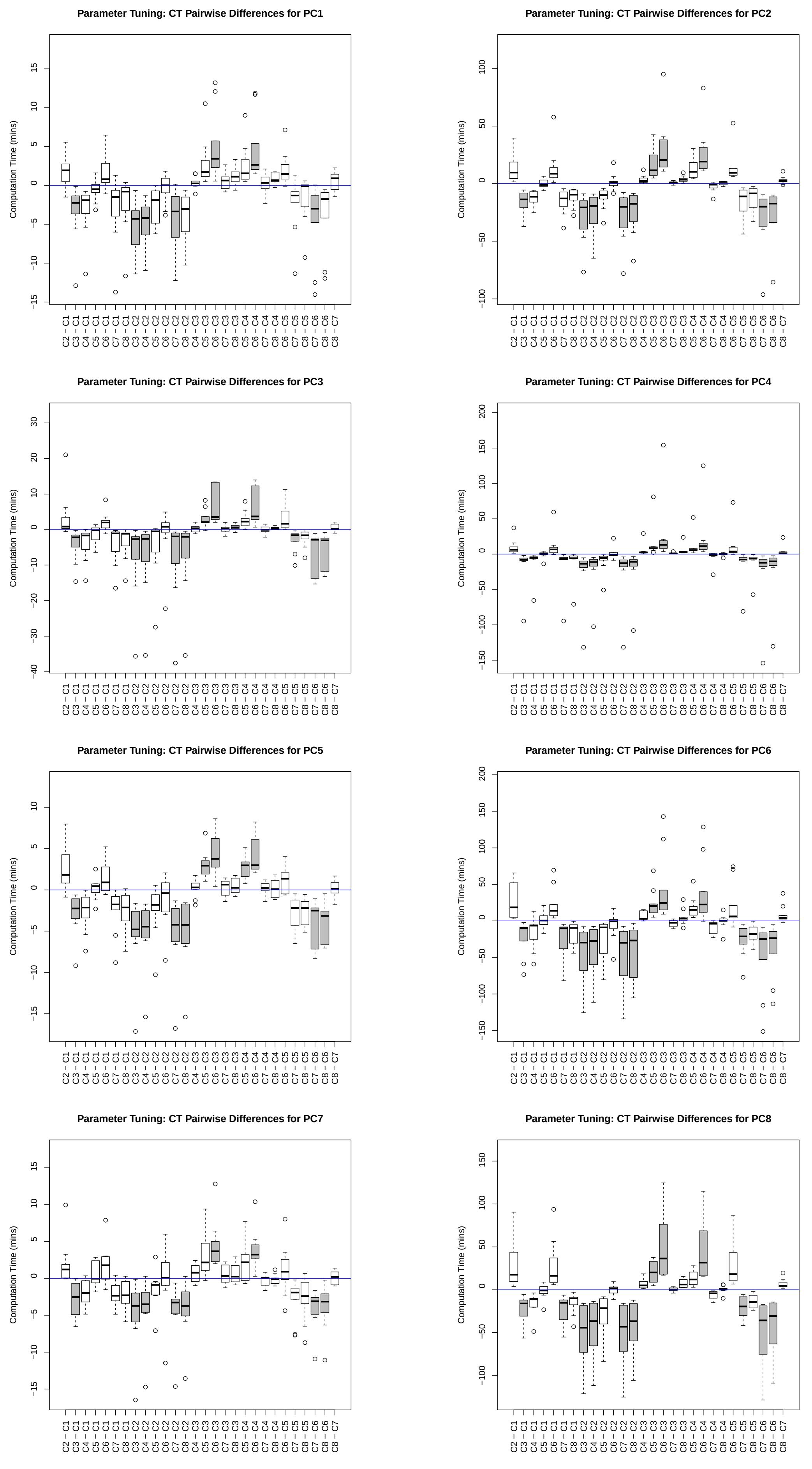


Figure A.1: Parameter Tuning: CT Pairwise Differences for PC1-PC8, C1-C8 (NB: y-axis scales vary, and grey boxes indicate significant differences at 0.05 level.)

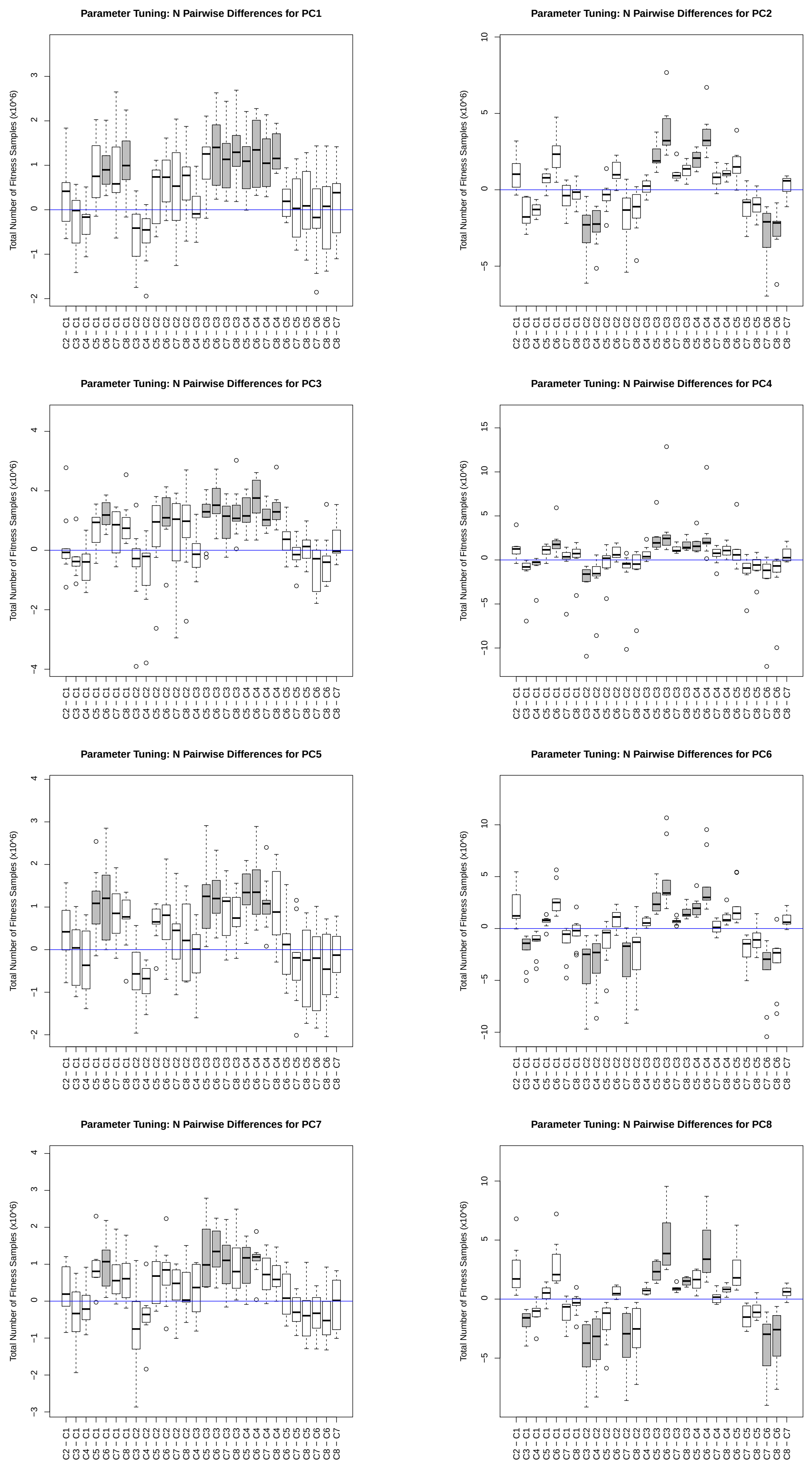


Figure A.2: Parameter Tuning: N Pairwise Differences for PC1-PC8, C1-C8 (NB: y-axis scales vary, and grey boxes indicate significant differences at 0.05 level.)

Table A.7: Parameter Tuning: Results for SQ, PC1-PC4, C4, C9-C11

	PC1		PC2		PC3		PC4	
	mean	sd	mean	sd	mean	sd	mean	sd
C4	0.6283052	0.2138481	0.2362546	0.1236281	0.6042334	0.3054539	0.3636693	0.1970668
C9	0.632594	0.2211151	0.2499705	0.1328568	0.6174854	0.3010965	0.3994749	0.1878298
C10	0.7002383	0.2346570	0.4118792	0.2110823	0.7482786	0.2629354	0.5420629	0.2101486
C11	0.7152585	0.2298299	0.4514993	0.2343482	0.7417048	0.2622576	0.5713693	0.2001278

Table A.8: Parameter Tuning: Results for SQ, PC5-PC8, C4, C9-C11

	PC5		PC6		PC7		PC8	
	mean	sd	mean	sd	mean	sd	mean	sd
C4	0.5456021	0.1750339	0.1131496	0.1366600	0.5973881	0.1922473	0.1131510	0.1280046
C9	0.5760188	0.1767313	0.160277	0.2135849	0.6230453	0.1640182	0.09478317	0.1077983
C10	0.7194111	0.1388876	0.2848848	0.2810054	0.8010506	0.1900822	0.2204204	0.2407496
C11	0.7687677	0.1376126	0.2299829	0.2095434	0.7831548	0.1857303	0.1941379	0.1984101

Table A.9: Parameter Tuning: Results for CT (milliseconds), PC1-PC4, C4, C9-C11

	PC1		PC2		PC3		PC4	
	mean	sd	mean	sd	mean	sd	mean	sd
C4	177465.3	64589.71	612275.6	373842.3	191977.7	80719.48	697703.3	949344
C9	243679.9	84756.6	862655.1	448800.6	270353.9	112792.3	1001112	1520365
C10	749583.6	481518	3785094	2702692	953549.3	879310.8	4629516	8785803
C11	997392.9	635634.6	4734667	4185259	1238702	1075688	6062606	12178240

Table A.10: Parameter Tuning: Results for CT (milliseconds), PC5-PC8, C4, C9-C11

	PC5		PC6		PC7		PC8	
	mean	sd	mean	sd	mean	sd	mean	sd
C4	236343.3	80372.76	1995638	2182651	240777.8	52267.79	1202544	640762.7
C9	313874.3	91602.17	2420904	2431685	321612.3	67919.88	1763364	1054049
C10	940181.4	323354.2	8414491	7115406	820540.3	238878.2	7332480	5131438
C11	1084080	444751.1	10939416	9612943	1116901	257383.6	8515249	5498971

Table A.11: Parameter Tuning: Results for N, PC1-PC4, C4, C9-C11

	PC1		PC2		PC3		PC4	
	mean	sd	mean	sd	mean	sd	mean	sd
C4	2341098	403077.9	2425277	483015.7	2235983	369770.8	2776157	1141218
C9	3745527	371923.1	4272486	515354.9	4089733	601186.5	4530089	1936193
C10	4130637	629382.6	5350761	978216.5	4377730	835313.1	5385021	3151112
C11	7602329	1142081	8389291	1598680	7758617	1456522	9045046	4391578

Table A.12: Parameter Tuning: Results for N, PC5-PC8, C4, C9-C11

	PC5		PC6		PC7		PC8	
	mean	sd	mean	sd	mean	sd	mean	sd
C4	2523214	589835.9	3042367	833657.6	2788673	398244.6	3037142	497190.5
C9	4027967	459434	4923219	1089817	4276891	370634.0	4781990	1019238
C10	4855378	923654	6536044	2136906	4723514	889218.1	6153633	1810397
C11	7533241	1156388	10242796	2794145	8480967	1604749	9947308	1834294

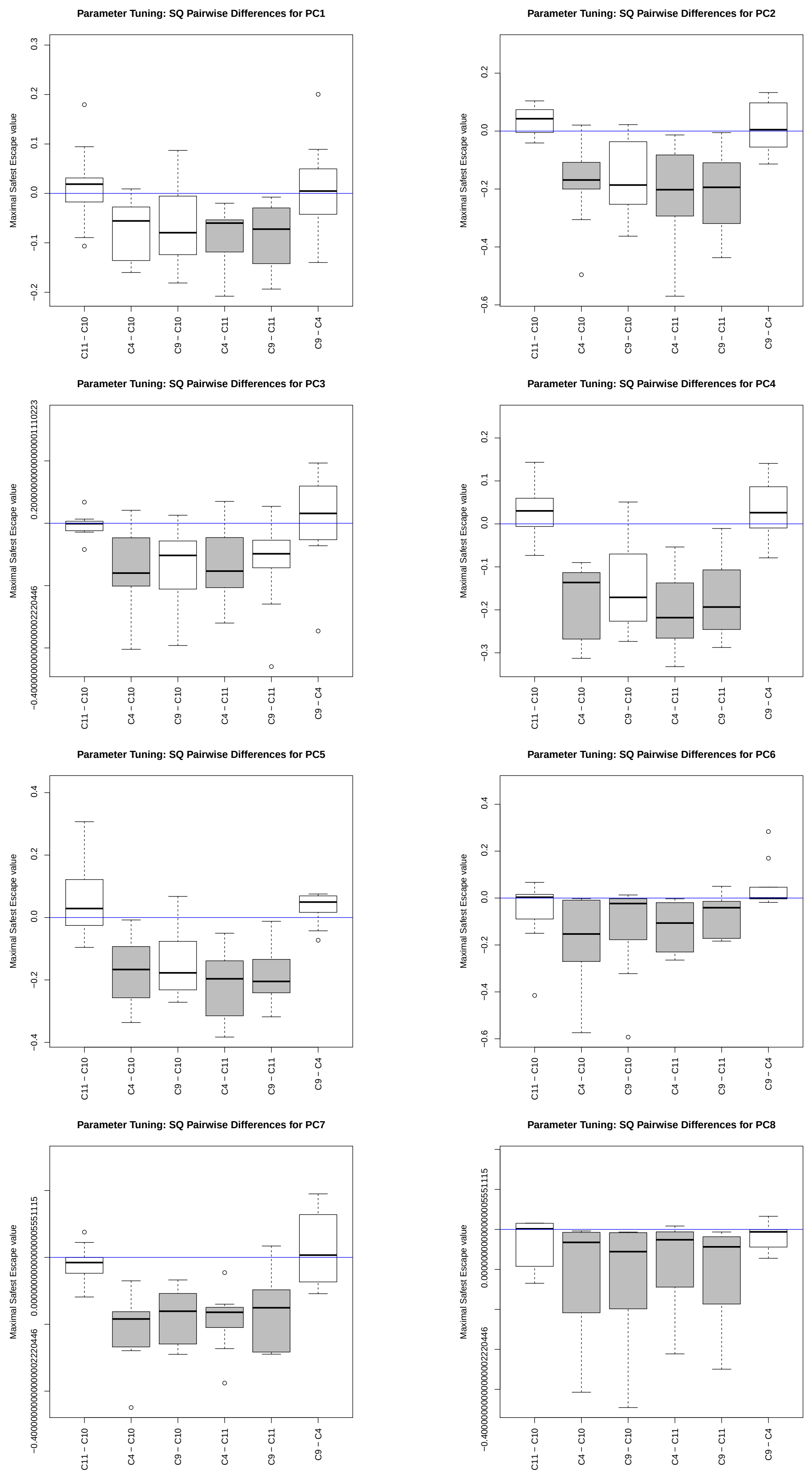


Figure A.3: Parameter Tuning: SQ Pairwise Differences for PC1-PC8, C4, C9-C11 (NB: y-axis scales vary, and grey boxes indicate significant differences at 0.05 level.)

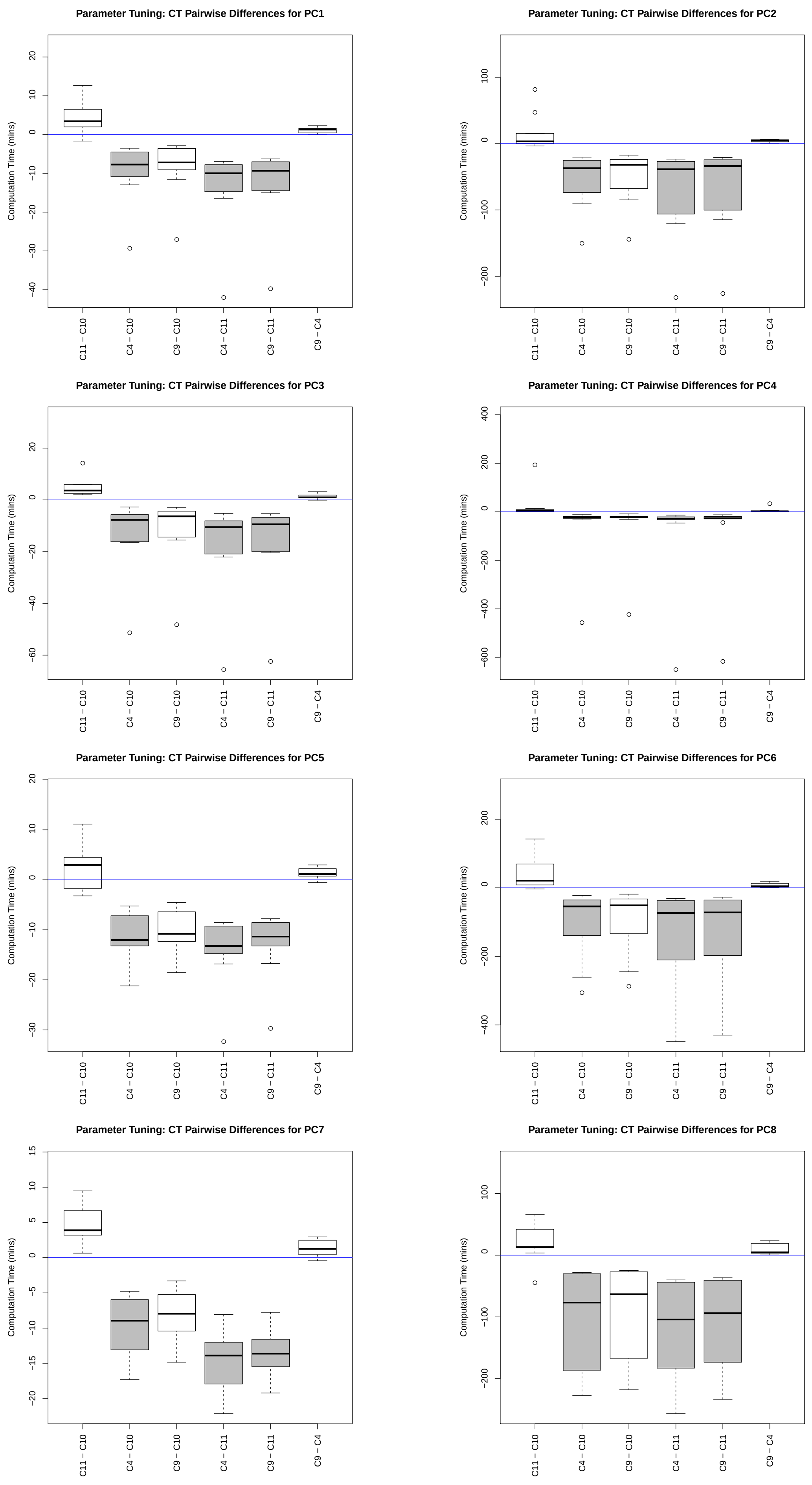


Figure A.4: Parameter Tuning: CT Pairwise Differences for PC1-PC8, C4, C9-C11 (NB: y-axis scales vary, and grey boxes indicate significant differences at 0.05 level.)

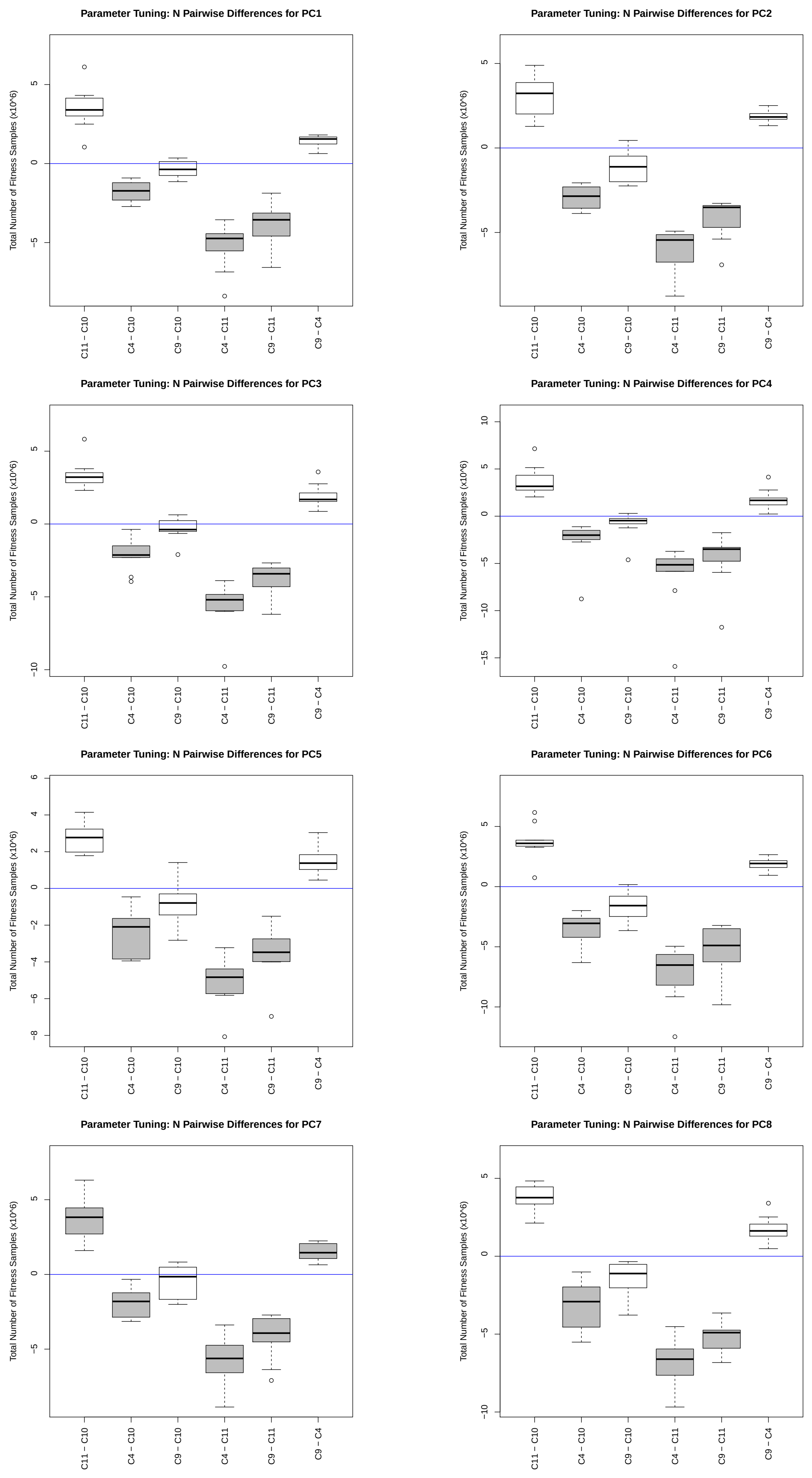


Figure A.5: Parameter Tuning: N Pairwise Differences for PC1-PC8, C4, C9-C11 (NB: y-axis scales vary, and grey boxes indicate significant differences at 0.05 level.)

The p-values for the post hoc pairwise comparison tests are provided in Tables A.19 and A.20 for solution quality, Tables A.21 and A.22 for computation time, and in Table A.23 for total number of samples.

A.2.1 Validation of Results

Boxplots of the results for the validation experiments are provided in Figures A.6 and A.7 for solution quality, Figure A.8 for computation time and Figure A.9 for total number of samples.

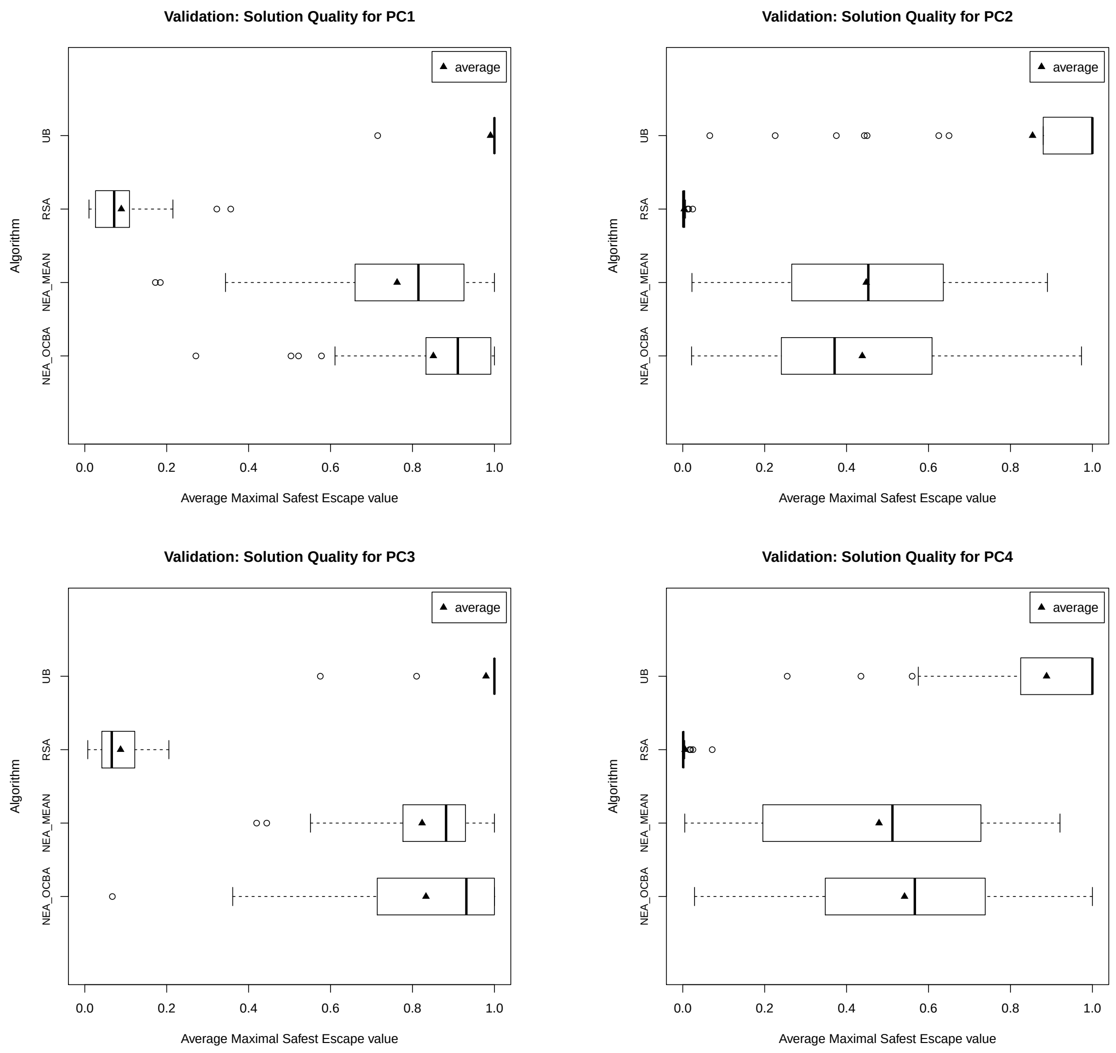


Figure A.6: Validation: SQ for PC1-PC4

Table A.13: Evaluation: Results for SQ, PC1-PC4

	PC1		PC2		PC3		PC4	
	mean	sd	mean	sd	mean	sd	mean	sd
NEA$_{OCBA}$	0.8403891	0.1640942	0.4222922	0.2535015	0.8411101	0.2025462	0.5594921	0.2491637
NEA$_{MEAN}$	0.745502	0.2147726	0.4648235	0.2736969	0.8355198	0.1552798	0.4560826	0.2849092
RSA	0.0839998	0.07931672	0.003881963	0.005441183	0.09881648	0.07448442	0.005687219	0.01370314
UB	0.9903333	0.05205659	0.8560756	0.2623151	0.9796667	0.07790949	0.8906667	0.1953490

Table A.14: Evaluation: Results for SQ, PC5-PC8

	PC5		PC6		PC7		PC8	
	mean	sd	mean	sd	mean	sd	mean	sd
NEA$_{OCBA}$	0.6768634	0.2261315	0.2702148	0.1964049	0.6795201	0.2328699	0.2517446	0.1794217
NEA$_{MEAN}$	0.6548175	0.2375635	0.2086036	0.1612185	0.7373045	0.1854246	0.2747924	0.2314703
RSA	0.03373751	0.058529	0.0007418709	0.001267179	0.02850247	0.03256121	0.0003559871	0.0007301304
UB	0.964	0.1147563	0.936225	0.1119413	0.9705	0.08561883	0.9145	0.2327099

Table A.15: Evaluation: Results for CT (ms), PC1-PC4

	PC1		PC2		PC3		PC4	
	mean	sd	mean	sd	mean	sd	mean	sd
NEA$_{OCBA}$	486514.6	213512.5	4633684	6814977	500257.4	255294.3	2445076	3039694
NEA$_{MEAN}$	182729.6	90168.03	278939.3	93383.97	151761.6	91224.74	253767.1	57269.44
RSA	865165.9	271529.6	2459375	1889210	815642.6	276932.4	1720793	585416.8
UB	7797.633	2658.89	8191.4	4224.069	14349.07	2051.522	14039.2	1318.821

Table A.16: Evaluation: Results for CT (ms), PC5-PC8

	PC5		PC6		PC7		PC8	
	mean	sd	mean	sd	mean	sd	mean	sd
NEA$_{OCBA}$	1042081	636109	6001180	7831690	918481.2	433929.1	4707376	3374709
NEA$_{MEAN}$	718221.8	2440992	773158.4	1566742	506653.8	1228852	492942.8	267098.7
RSA	2738694	3233622	3610715	3202175	2049538	1700005	4094953	3882433
UB	72682.73	189627.5	16977.53	21439.96	105383.2	315135.7	117848.6	475982.8

Table A.17: Evaluation: Results for N, PC1-PC4

	PC1		PC2		PC3		PC4	
	mean	sd	mean	sd	mean	sd	mean	sd
NEA$_{OCBA}$	3671437	949683	4803123	2547565	3863930	971678.8	4055276	1259626
NEA$_{MEAN}$	3005601	1142.869	3005481	1144.731	3005940	323.1099	3005444	1006.580
RSA	3372965	127008.0	3802778	635288	3324309	140970.9	3680207	267514.2

Table A.18: Evaluation: Results for N, PC5-PC8

	PC5		PC6		PC7		PC8	
	mean	sd	mean	sd	mean	sd	mean	sd
NEA$_{OCBA}$	3420241	509270.4	5144032	2771460	3519799	649807.3	4861685	1385640
NEA$_{MEAN}$	3005858	378.0423	3005683	890.2702	3006000	0	3005599	952.5802
RSA	3467757	233535.5	3923132	539330.3	3398444	188744.3	3881064	497061.5

PC1	NEA$_{MEAN}$	NEA$_{OCBA}$	RSA
NEA$_{OCBA}$	**0.02061903720611689**	NA	NA
RSA	**0.00000000058503906**	**0.00000000014995503**	NA
UB	**0.00000000060156998**	**0.00000000534863786**	**0.00000000001031331**

PC2	NEA$_{MEAN}$	NEA$_{OCBA}$	RSA
NEA$_{OCBA}$	0.70819305290442380	NA	NA
RSA	**0.00000000014010457**	**0.00000000014010457**	NA
UB	**0.00000093372381656**	**0.00000045537504267**	**0.00000000007030262**

PC3	NEA$_{MEAN}$	NEA$_{OCBA}$	RSA
NEA$_{OCBA}$	0.46390014392935219	NA	NA
RSA	**0.00000000015080377**	**0.00000000052533769**	NA
UB	**0.00000000972927770**	**0.00000015964037916**	**0.00000000001419404**

PC4	NEA$_{MEAN}$	NEA$_{OCBA}$	RSA
NEA$_{OCBA}$	0.18717863983208879	NA	NA
RSA	**0.00000000022857021**	**0.00000000016507769**	NA
UB	**0.00000016796634183**	**0.00000118071643386**	**0.00000000006753308**

Table A.19: Evaluation: P-values for post hoc pairwise comparison tests, SQ, PC1-PC4. (Bold indicates significant result at 0.05 level.)

PC5	NEA_{MEAN}	NEA_{OCBA}	RSA
NEA_{OCBA}	0.66273036217571124	NA	NA
RSA	**0.00000000020297707**	**0.00000000020297707**	NA
UB	**0.00000000158288245**	**0.00000000463567309**	**0.00000000001887531**

PC6	NEA_{MEAN}	NEA_{OCBA}	RSA
NEA_{OCBA}	0.35810141979272925	NA	NA
RSA	**0.00000000007898858**	**0.00000000007898858**	NA
UB	**0.00000000007898858**	**0.00000000007898858**	**0.00000000004340424**

PC7	NEA_{MEAN}	NEA_{OCBA}	RSA
NEA_{OCBA}	0.40340685182518909	NA	NA
RSA	**0.00000000015023746**	**0.00000000015023746**	NA
UB	**0.00000003303927420**	**0.00000003929785358**	**0.00000000003122322**

PC8	NEA_{MEAN}	NEA_{OCBA}	RSA
NEA_{OCBA}	0.947355822930050606	NA	NA
RSA	**0.000000000039089949**	**0.000000000039089949**	NA
UB	**0.000000001573156424**	**0.000000001573156424**	**0.000000000008279932**

Table A.20: Evaluation: P-values for post hoc pairwise comparison tests, SQ, PC5-PC8.
(Bold indicates significant result at 0.05 level.)

PC1	NEA$_{MEAN}$	NEA$_{OCBA}$	RSA
NEA$_{OCBA}$	**0.0000000000126680700187**	NA	NA
RSA	**0.000000000000000102**	**0.0000000342959129633882**	NA
UB	**0.000000000000000102**	**0.000000000000000102**	**0.000000000000000102**

PC2	NEA$_{MEAN}$	NEA$_{OCBA}$	RSA
NEA$_{OCBA}$	**0.000000000000000102**	NA	NA
RSA	**0.000000000000000102**	0.1381260880781920441507	NA
UB	**0.000000000000000102**	**0.000000000000000102**	**0.000000000000000102**

PC3	NEA$_{MEAN}$	NEA$_{OCBA}$	RSA
NEA$_{OCBA}$	**0.0000000000009683034302**	NA	NA
RSA	**0.000000000000000102**	**0.0000029128393119885165**	NA
UB	**0.000000000000000102**	**0.000000000000000102**	**0.000000000000000102**

PC4	NEA$_{MEAN}$	NEA$_{OCBA}$	RSA
NEA$_{OCBA}$	**0.0000000000000001014674**	NA	NA
RSA	**0.000000000000000102**	0.3980033437616471991838	NA
UB	**0.000000000000000102**	**0.000000000000000102**	**0.000000000000000102**

Table A.21: Evaluation: P-values for post hoc pairwise comparison tests, CT, PC1-PC4.
(Bold indicates significant result at 0.05 level.)

PC5	NEA$_{MEAN}$	NEA$_{OCBA}$	RSA
NEA$_{OCBA}$	**0.000000000596917716749**	NA	NA
RSA	**0.000000000001936606860**	**0.008383152587827669838**	NA
UB	**0.000000002859009109287**	**0.000000000001246527050**	**0.000000000000009842338**

PC6	NEA$_{MEAN}$	NEA$_{OCBA}$	RSA
NEA$_{OCBA}$	**0.000000000017994736648**	NA	NA
RSA	**0.000000000365075320343**	**0.0243354659261669839521**	NA
UB	**0.000000000000000102**	**0.000000000000000102**	**0.000000000000000102**

PC7	NEA$_{MEAN}$	NEA$_{OCBA}$	RSA
NEA$_{OCBA}$	**0.000000006260284171**	NA	NA
RSA	**0.000000000008374697**	**0.001092004716304495**	NA
UB	**0.000000000203928984**	**0.000000000014526073**	**0.000000000000943241**

PC8	NEA$_{MEAN}$	NEA$_{OCBA}$	RSA
NEA$_{OCBA}$	**0.000000000000000102**	NA	NA
RSA	**0.000000000000000102**	0.1193576070154743207219	NA
UB	**0.000000000009683034302**	**0.00000000000000656**	**0.0000000000000189**

Table A.22: Evaluation: P-values for post hoc pairwise comparison tests, CT, PC5-PC8. (Bold indicates significant result at 0.05 level.)

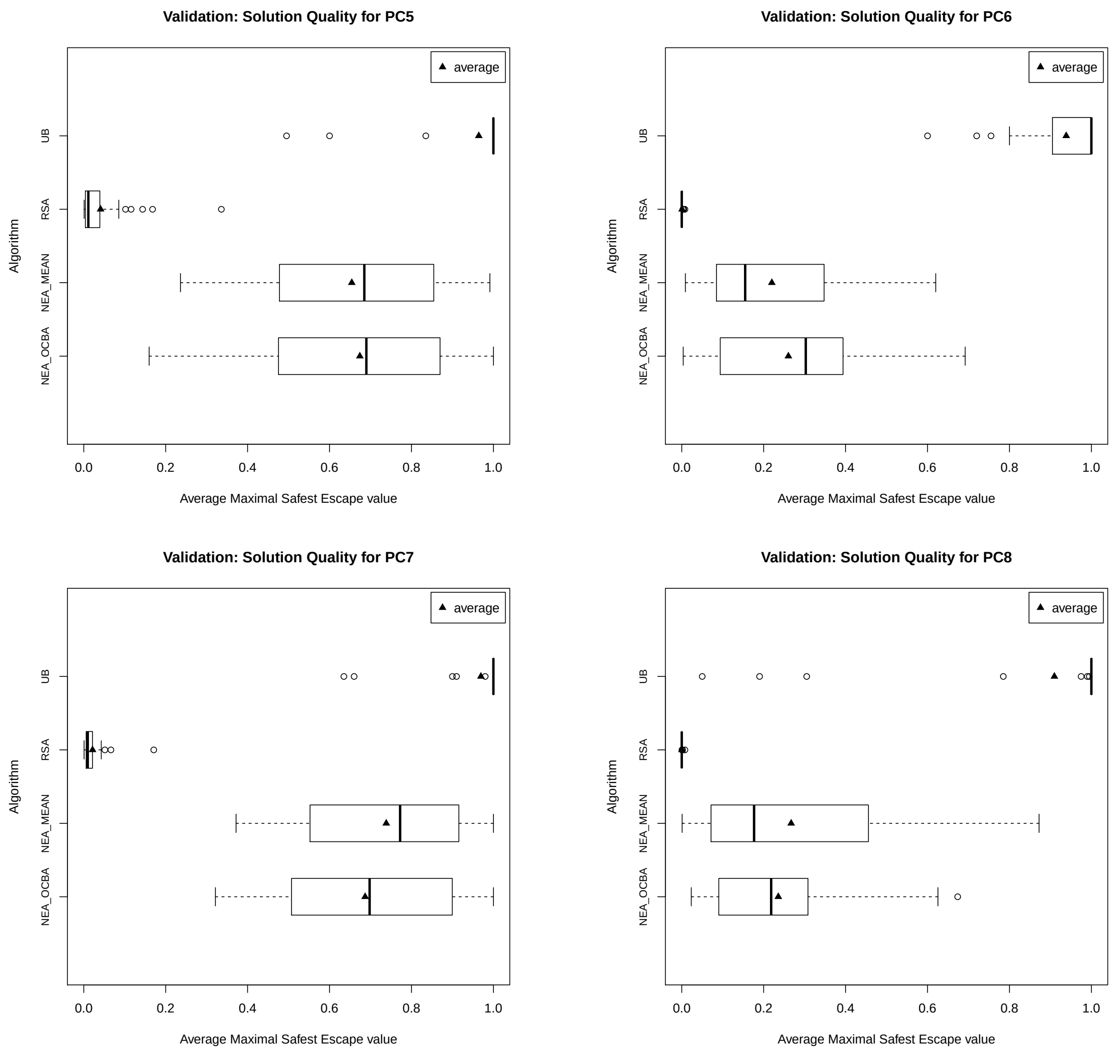


Figure A.7: Validation: SQ for PC5-PC8

A.2.2 Evaluation of NEA$_{OCBA}$ with HSEA

The following boxplots show the results of the HSEA and NEA$_{OCBA}$ for solution quality, in Figures A.10 and A.11, and computation time, in FigureA.12, for each problem class.

Table A.24 provides the p-values from the hypothesis tests comparing the NEA$_{OCBA}$ results with the HSEA.

Figure A.13 shows the differences between the NEA$_{OCBA}$ and the HSEA for solution quality and computation time, respectively.

A.3 Discussion

Boxplots of the results for the additional experiments are provided in Figures A.14 and A.15 for solution quality, Figure A.16 for computation time and Figure A.17 for total number of samples.

The p-values for the hypothesis tests between the NEA$^{NS}_{OCBA}$ and the NEA$^{NS}_{MEAN}$ are provided in Table A.25. The p-values for the hypothesis tests between the NEA$_{OCBA}$ and the NEA$^{NS}_{OCBA}$, and the NEA$_{MEAN}$ and the NEA$^{NS}_{MEAN}$ are provided in Table A.26 and Table A.27, respectively.

Boxplots showing the differences between the NEA$_{OCBA}$ and the NEA$^{NS}_{OCBA}$, and between the NEA$_{MEAN}$ and the NEA$^{NS}_{MEAN}$ are shown in Figures A.18 and A.19, respectively.

PC1	NEA$_{MEAN}$	NEA$_{OCBA}$
NEA$_{OCBA}$	**0.00000000001563434**	NA
RSA	**0.00000000001563434**	**0.00831191165588153**

PC2	NEA$_{MEAN}$	NEA$_{OCBA}$
NEA$_{OCBA}$	**0.00000000001943684**	NA
RSA	**0.00000000001943684**	**0.01687053334542116**

PC3	NEA$_{MEAN}$	NEA$_{OCBA}$
NEA$_{OCBA}$	**0.000000000005160753**	NA
RSA	**0.000000000005160753**	0.561874772423794244

PC4	NEA$_{MEAN}$	NEA$_{OCBA}$
NEA$_{OCBA}$	**0.0000000000381038**	NA
RSA	**0.0000000000381038**	0.2664685034622974

PC5	NEA$_{MEAN}$	NEA$_{OCBA}$
NEA$_{OCBA}$	**0.00000000001233292**	NA
RSA	**0.00000000001233292**	**0.02531696007060259**

PC6	NEA$_{MEAN}$	NEA$_{OCBA}$
NEA$_{OCBA}$	**0.00000000002818083**	NA
RSA	**0.00000000002818083**	**0.00288294818758070**

PC7	NEA$_{MEAN}$	NEA$_{OCBA}$
NEA$_{OCBA}$	**0.000000000003635341**	NA
RSA	**0.000000000003635341**	0.328034557351709111

PC8	NEA$_{MEAN}$	NEA$_{OCBA}$
NEA$_{OCBA}$	**0.00000000001943684**	NA
RSA	**0.00000000001943684**	**0.00000868335417622**

Table A.23: Evaluation: P-values for post hoc pairwise comparison tests, N, PC1-PC8. (Bold indicates significant result at 0.05 level.)

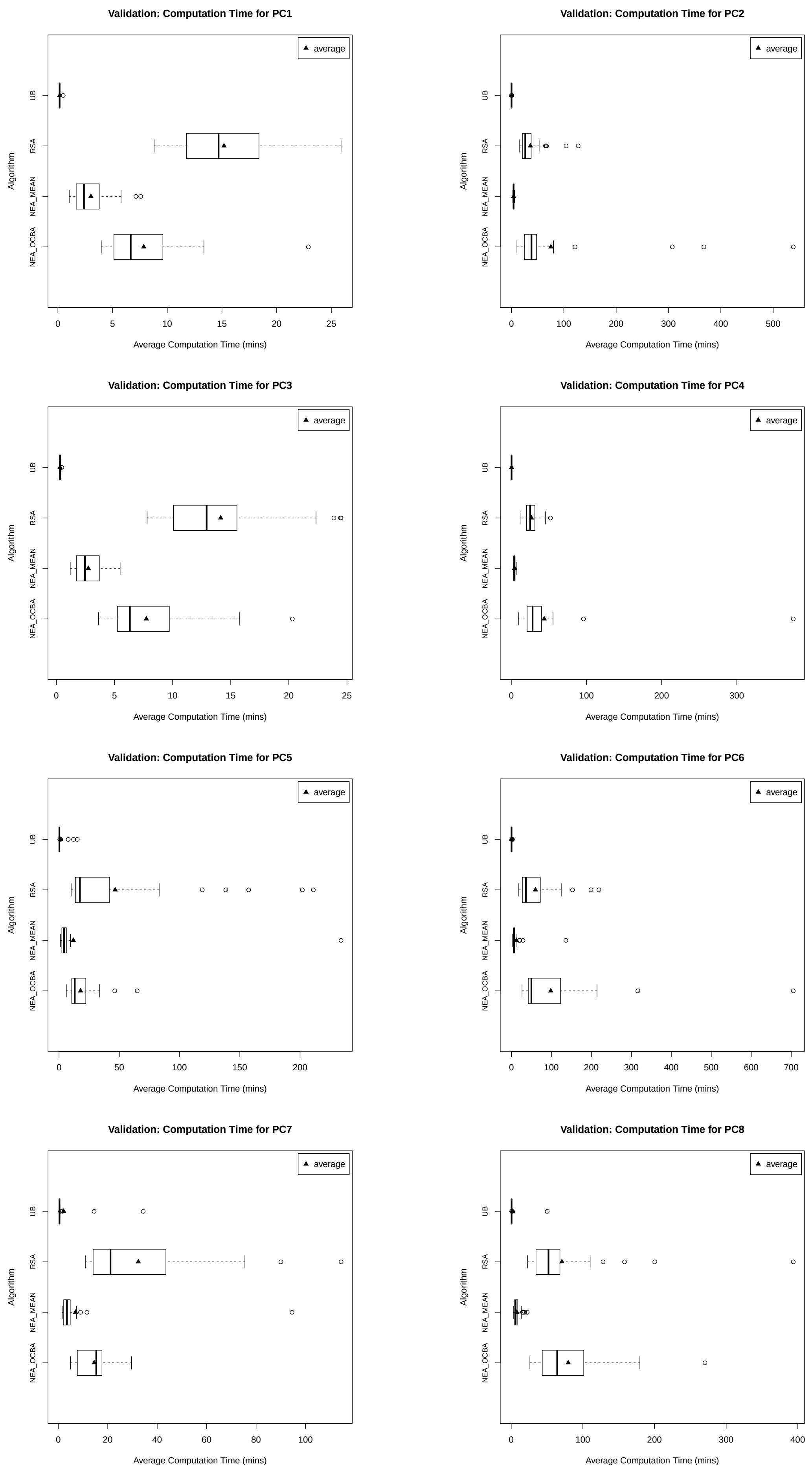


Figure A.8: Validation: CT for PC1-PC8

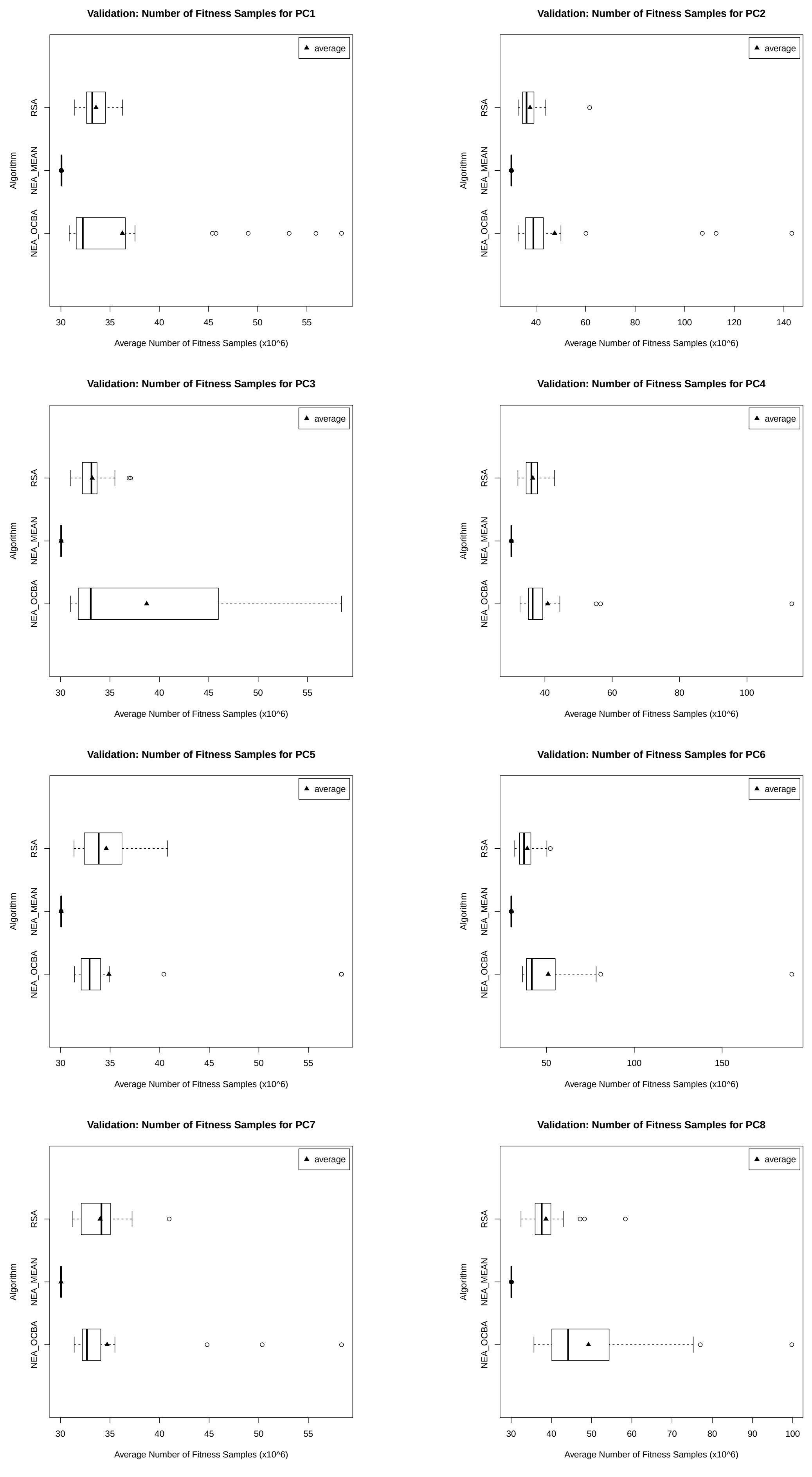


Figure A.9: Validation: N for PC1-PC8

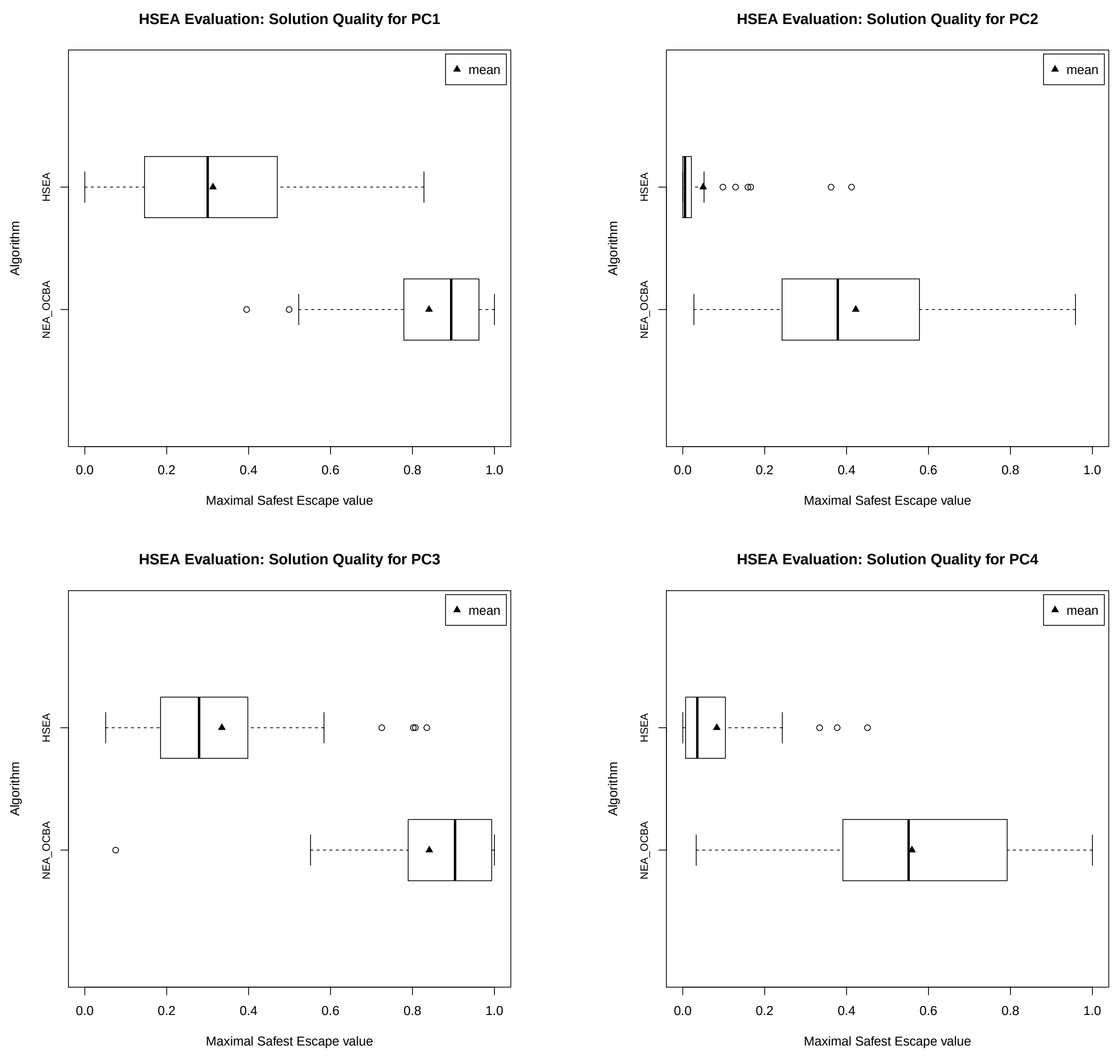


Figure A.10: HSEA Evaluation: SQ for PC1-PC4

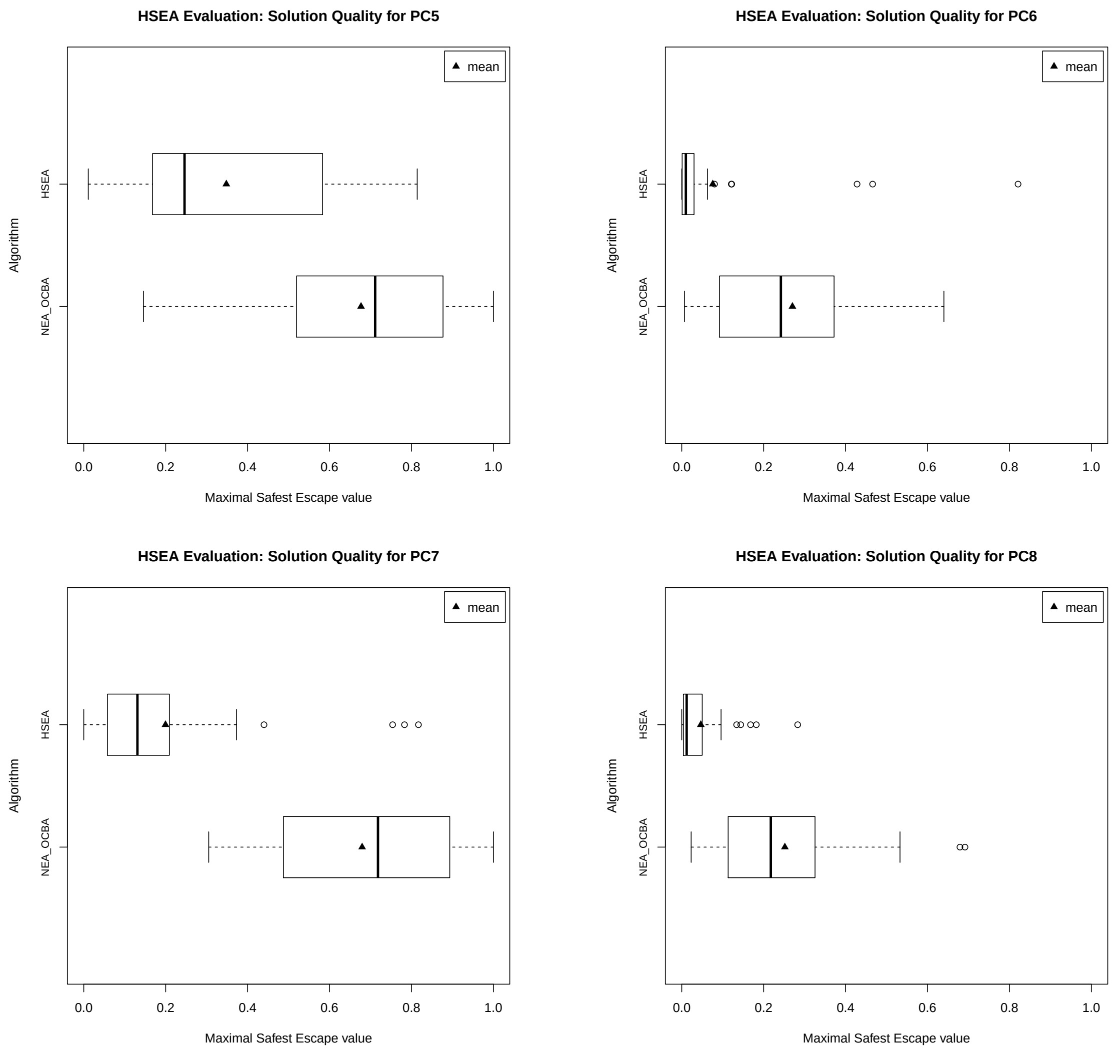


Figure A.11: HSEA Evaluation: SQ for PC5-PC8

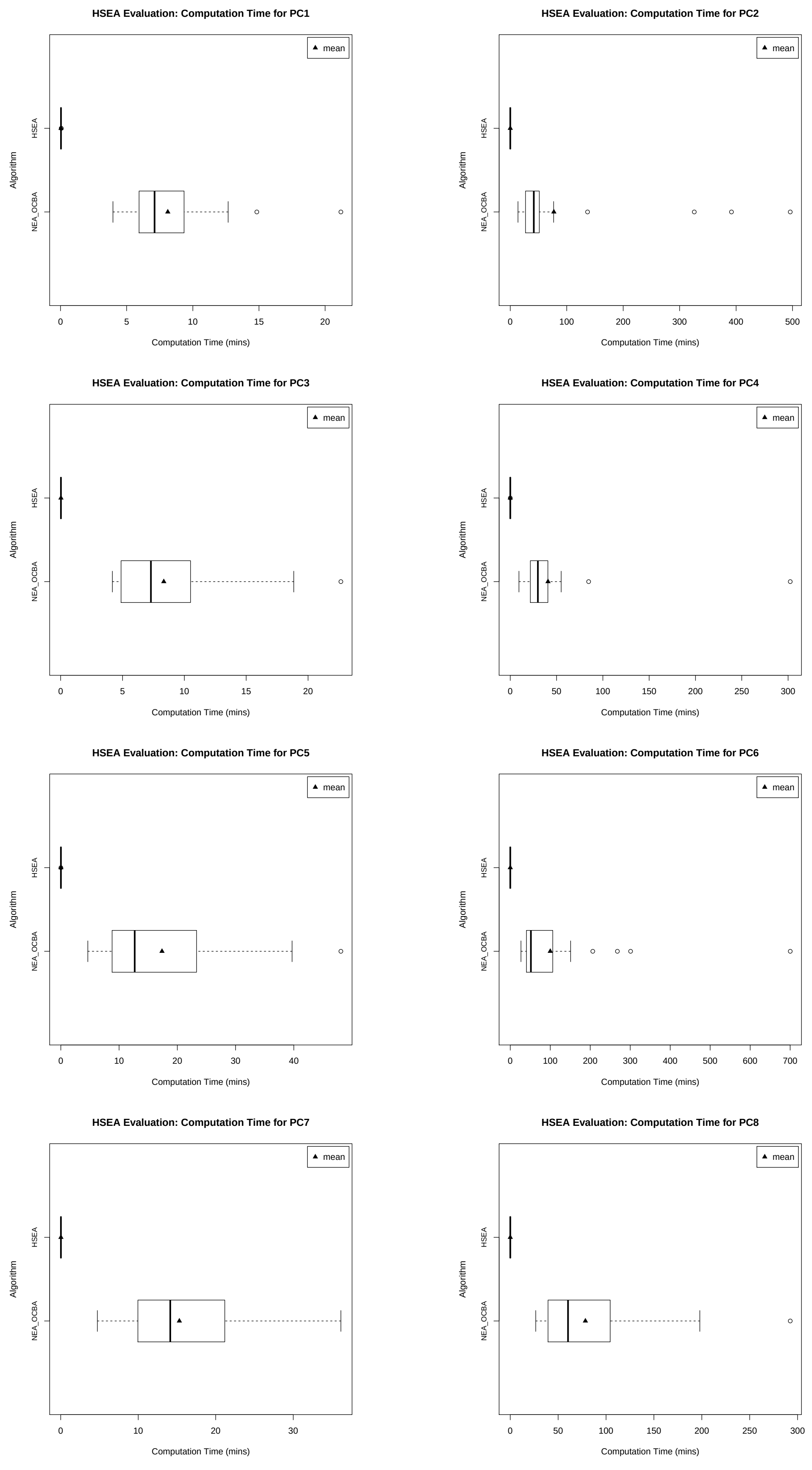


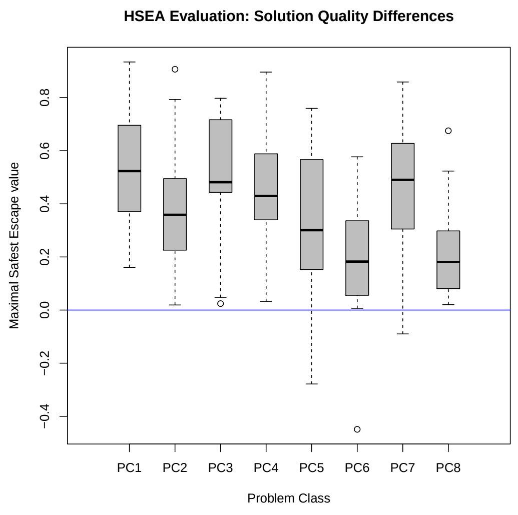
Figure A.12: HSEA Evaluation: CT for PC1-PC8

PC	SQ	CT
1	**0.0000000009313226**	**0.0000000009313226**
2	**0.0000000009313226**	**0.0000000009313226**
3	**0.0000000009313226**	**0.0000000009313226**
4	**0.0000000009313226**	**0.0000000009313226**
5	**0.0000008419156**	**0.0000000009313226**
6	**0.0000009955838**	**0.0000000009313226**
7	**0.000000001862645**	**0.0000000009313226**
8	**0.0000000009313226**	**0.0000000009313226**

Table A.24: HSEA Evaluation: P-values for hypothesis tests between the HSEA and NEA$_{OCBA}$. (Bold indicates significant result at 0.05 level.)

PC	SQ	CT	N
1	0.05744915	**0.00000000000000001691123**	**0.0000000000001720251**
2	**0.005437617**	**0.00000000000000001691123**	**0.00000000000121178**
3	0.06566001	**0.00000000000000001691123**	**0.0000000000001720251**
4	**0.01694876**	**0.00000000000000003382247**	**0.00000000000121178**
5	0.5641876	**0.00000000000000001691123**	**0.00000000000121178**
6	**0.00000002172021**	**0.00000000000000006764494**	**0.00000000000121178**
7	0.5996337	**0.00000000000000001691123**	**0.0000000000001210791**
8	**0.0000006352674**	**0.00000000000000001691123**	**0.00000000000121178**

Table A.25: Discussion: P-values for hypothesis tests between NEA$^{NS}_{OCBA}$ and NEA$^{NS}_{MEAN}$. (Bold indicates significant result at 0.05 level.)



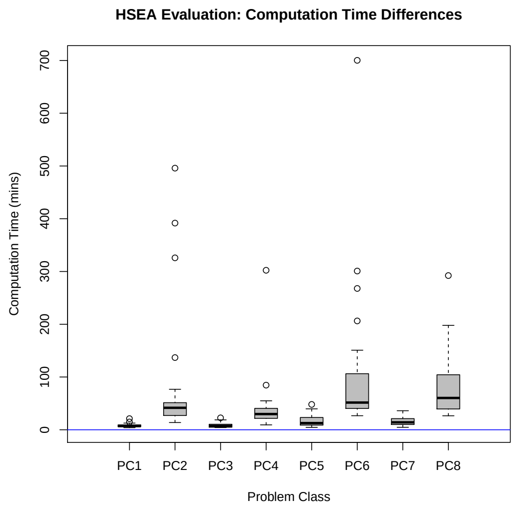
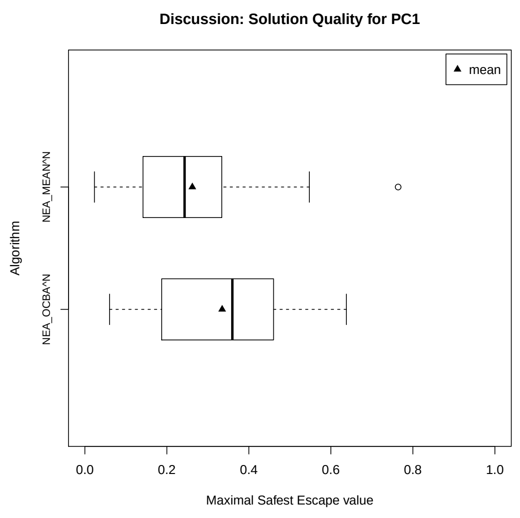
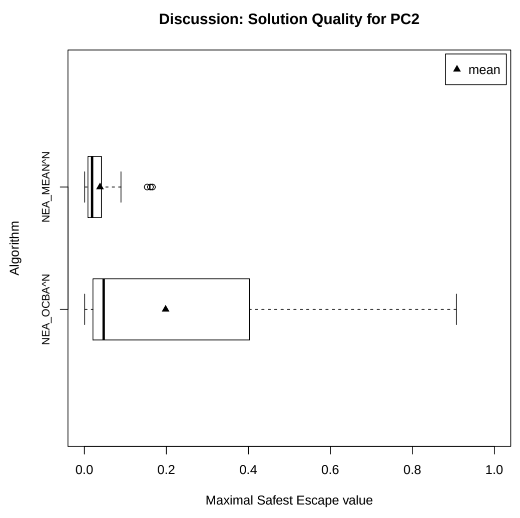
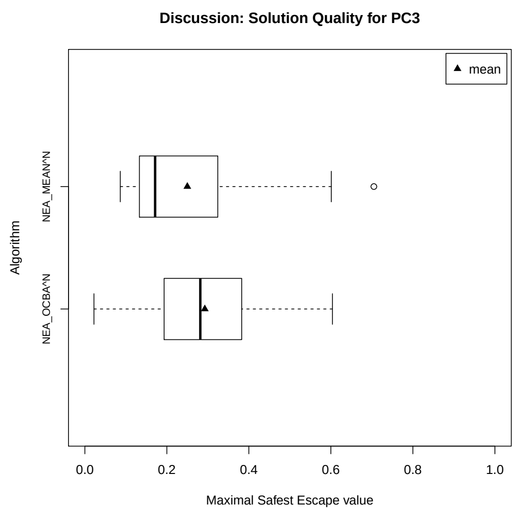


Figure A.13: HSEA Evaluation: Differences between HSEA and NEA$_{OCBA}$ (N.B. Grey boxes indicate significant results at 0.05 level.)







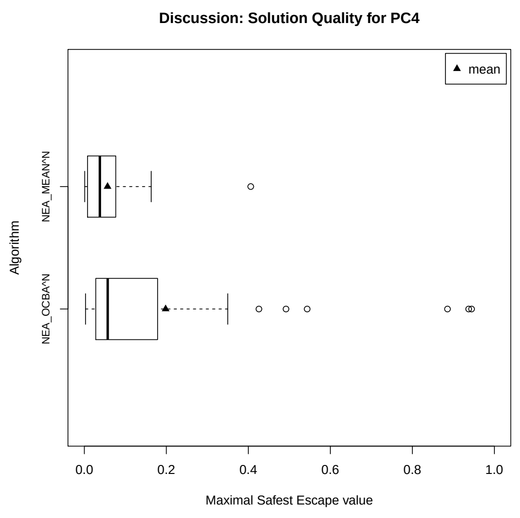


Figure A.14: Discussion: SQ for PC1-PC4

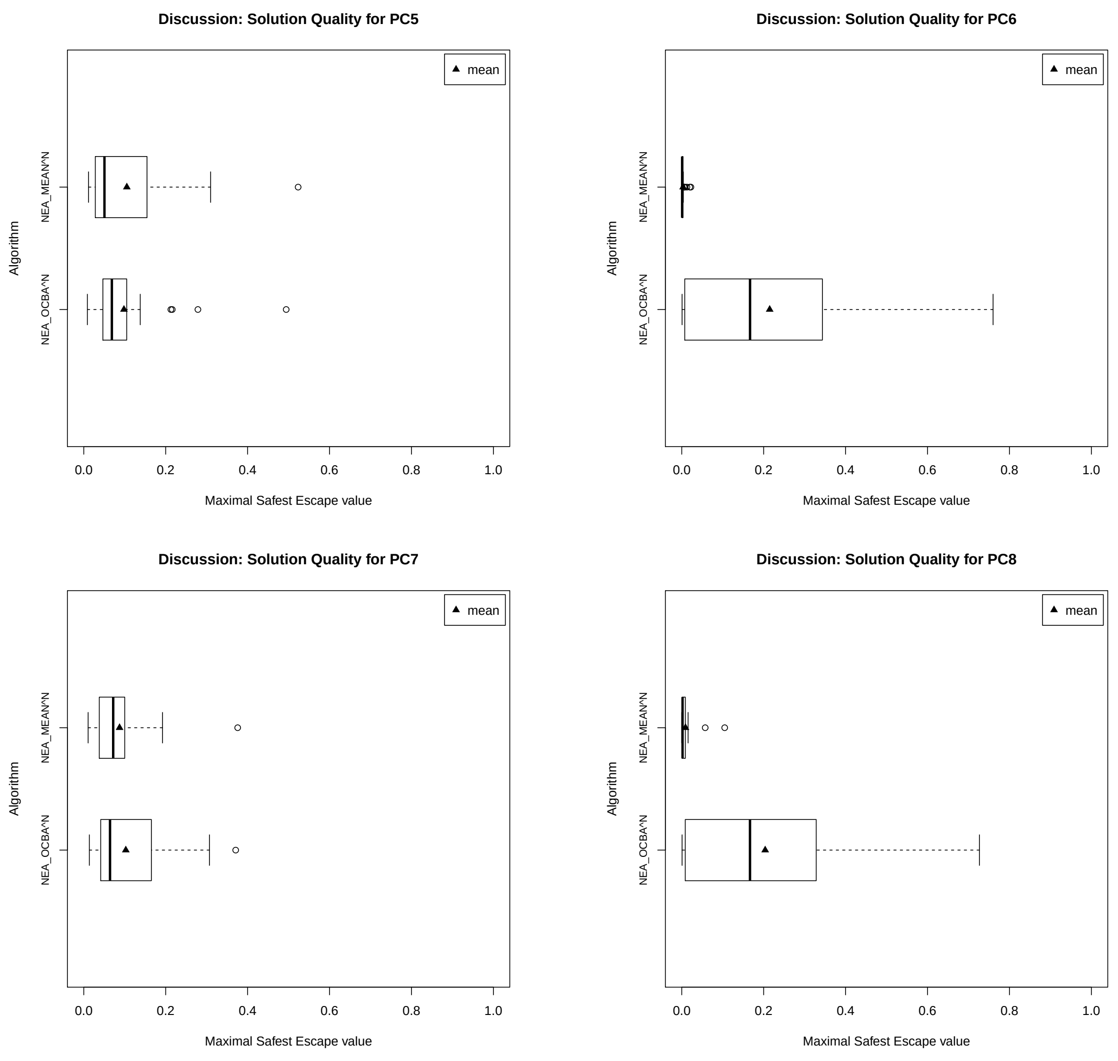
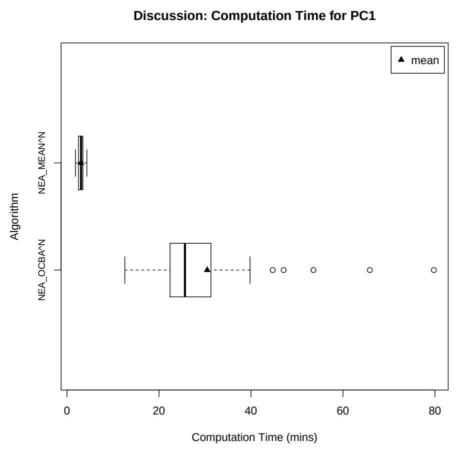
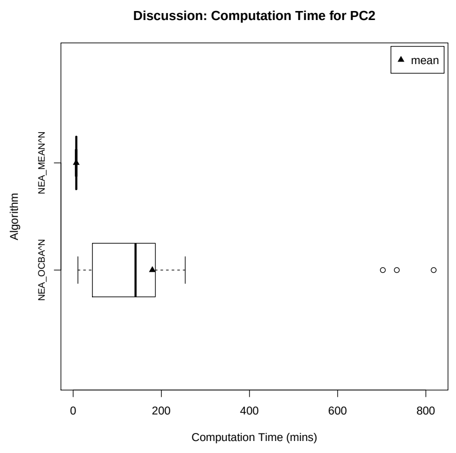
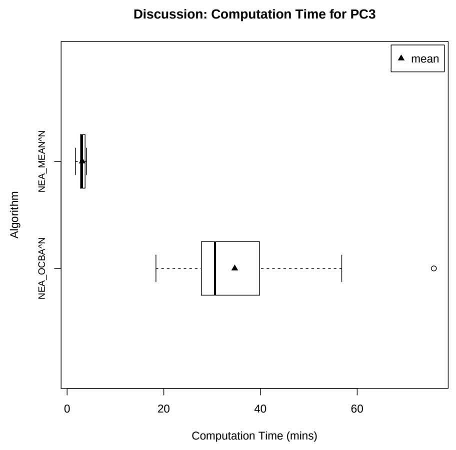
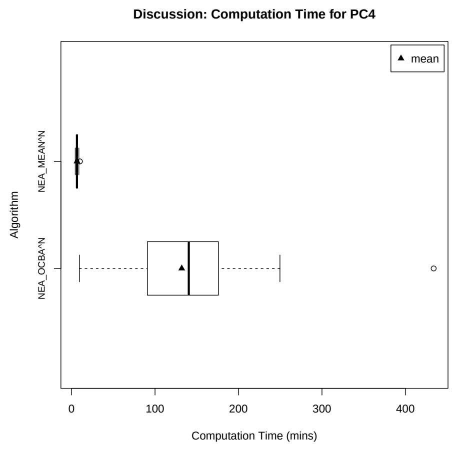


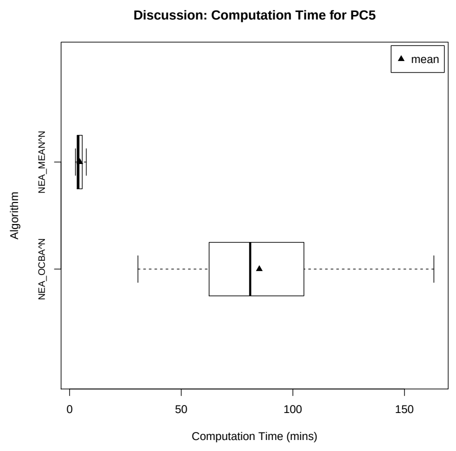
Figure A.15: Discussion: SQ for PC5-PC8

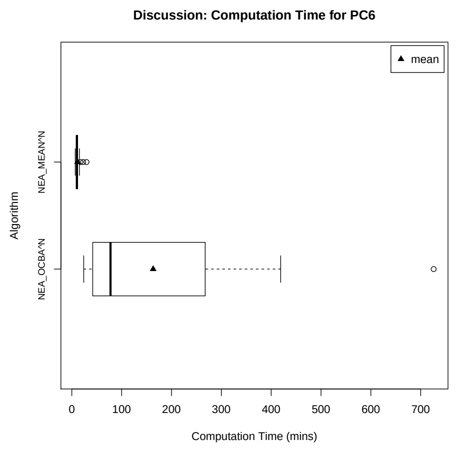


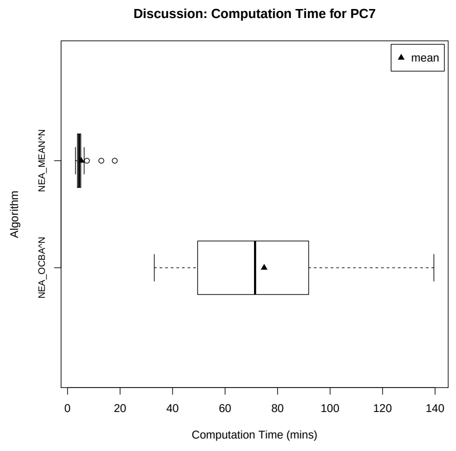












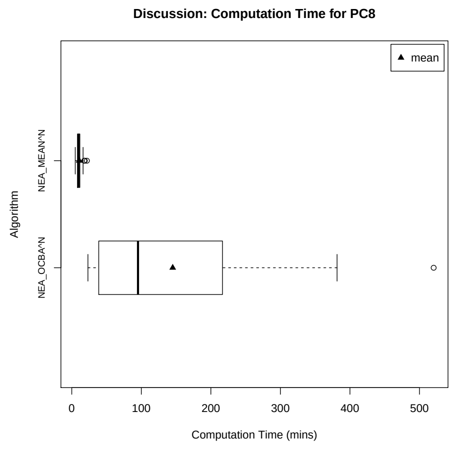


Figure A.16: Discussion: CT for PC1-PC8

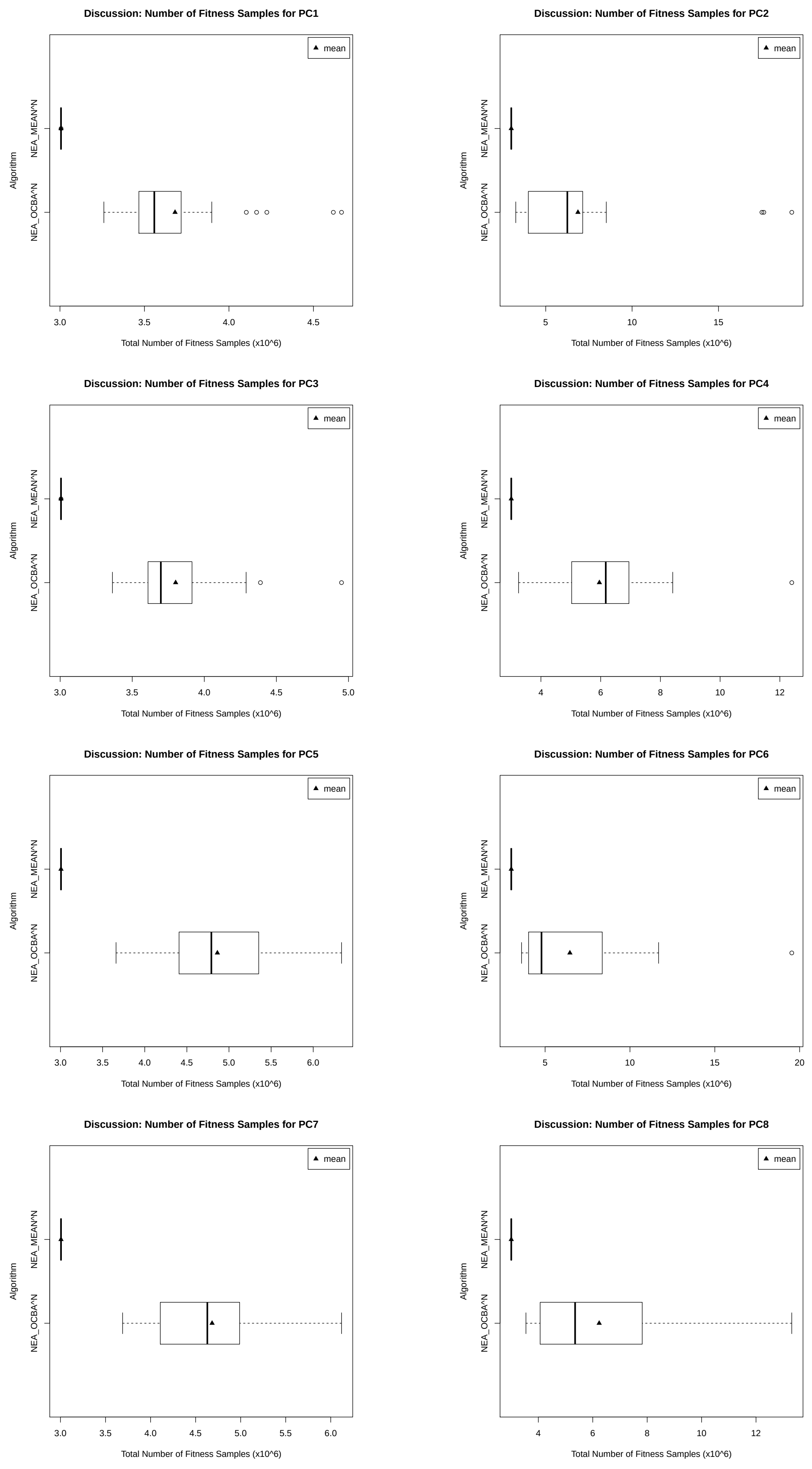


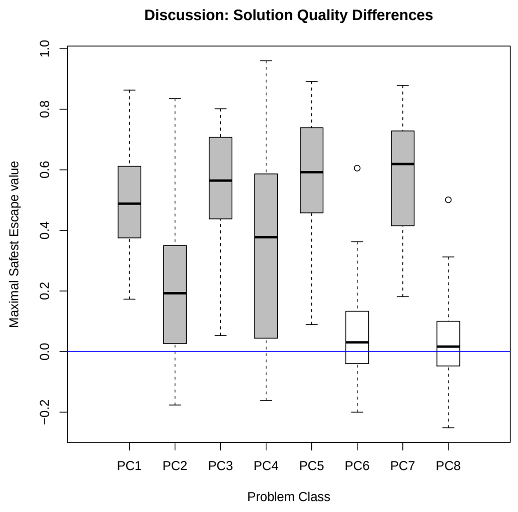
Figure A.17: Discussion: N for PC1-PC8

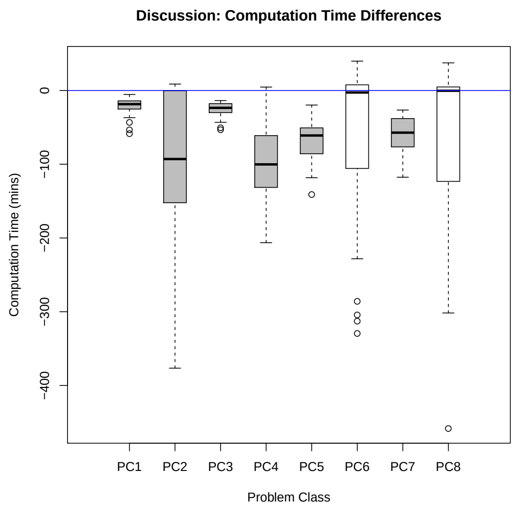
PC	SQ	CT	N
1	**0.000000001862645**	**0.000000001862645**	0.140283
2	**0.00006286614**	**0.00004407577**	**0.0000207983**
3	**0.000000001862645**	**0.000000001862645**	0.5561133
4	**0.000001683831**	**0.0000005718321**	**0.0000005718321**
5	**0.000000001862645**	**0.000000001862645**	**0.000001991168**
6	0.1003974	0.1094321	0.1003974
7	**0.000000001862645**	**0.000000001862645**	**0.00007910654**
8	0.2449464	0.08032736	**0.03453673**

Table A.26: Discussion: P-values for hypothesis tests between the NEA$_{OCBA}$ and the NEA$^{NS}_{OCBA}$. (Bold indicates significant result at 0.05 level.)

PC	SQ	CT	N
1	**0.00000000372529**	0.8078304	**0.03603169**
2	**0.000000001862645**	**0.000000314787**	**0.03603169**
3	**0.000000001862645**	**0.02209885**	0.3710934
4	**0.000000001862645**	**0.000000001862645**	**0.00588927**
5	**0.000000001862645**	0.9515265	0.1003482
6	**0.000000001862645**	**0.0001886003**	**0.0141474**
7	**0.000000001862645**	0.9515265	–
8	**0.000000001862645**	**0.0000002551824**	**0.03603169**

Table A.27: Discussion: P-values for hypothesis tests between the NEA$_{MEAN}$ and the NEA$^{NS}_{MEAN}$. (Bold indicates significant result at 0.05 level. Hyphen indicates identical results.)





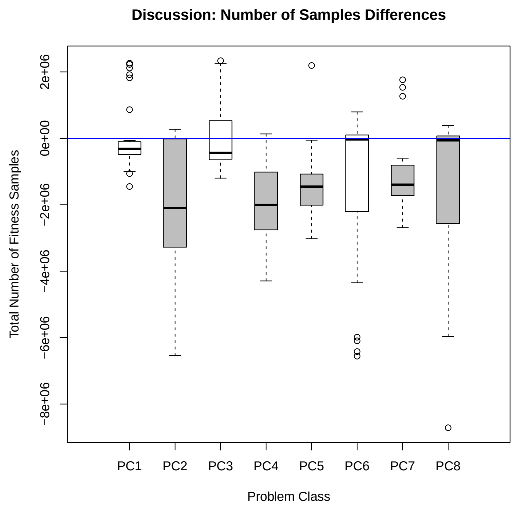
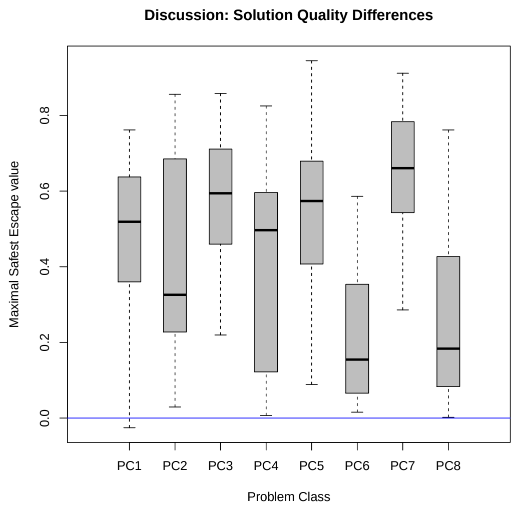
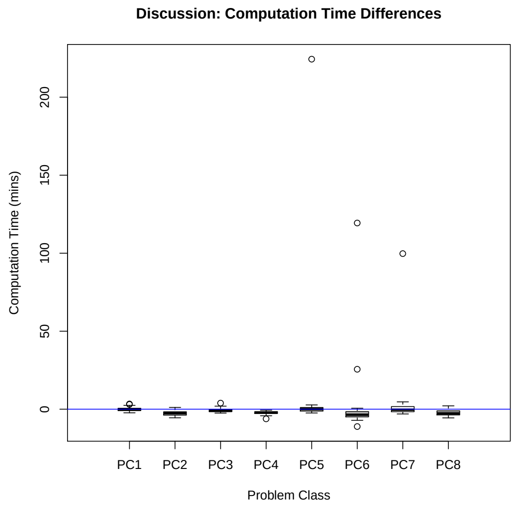


Figure A.18: Discussion: Differences between NEA$_{OCBA}$ and NEA$^{NS}_{OCBA}$ (N.B. Grey boxes indicate significant results at 0.05 level.)





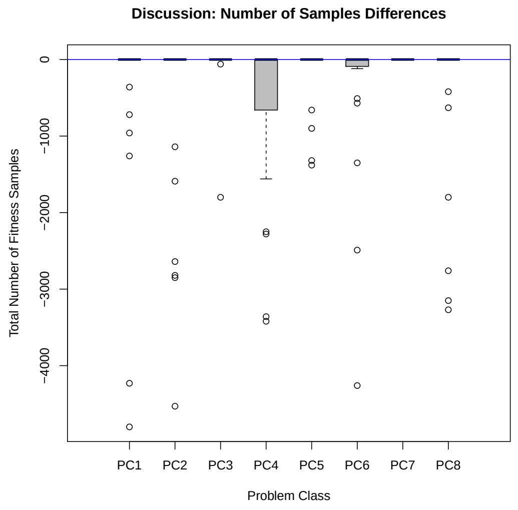


Figure A.19: Discussion: Differences between NEA$_{MEAN}$ and NEA$^{NS}_{MEAN}$ (N.B. Grey boxes indicate significant results at 0.05 level.)

Bibliography

C. Ahn and R. Ramakrishna. A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations. *IEEE Transactions on Evolutionary Computation*, 6 (6):566–579, December 2002.

R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Inc, 1993.

A. Aizawa and B. Wah. Dynamic control of genetic algorithms in a noisy environment. In *Proceedings of the Conference on Genetic Algorithms*, pages 48–55, 1993.

A. Aizawa and B. Wah. Scheduling of Genetic Algorithms in a Noisy Environment. *Evolutionary Computation*, 2(2), 1994.

E. Anderson, P. Nash, and A. Philpott. A Class of Continuous Network Flow Problems. *Mathematics of Operations Research*, 7(4):501–514, November 1982.

J. Aronson. A survey of dynamic network flows. *Annals of Operations Research*, 20: 1–66, 1989.

T. Bäck, D. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Institute of Physics Publishing Ltd., Bristol, UK, 1997.

J. Baker. Reducing bias and inefficiency in the selection algorithm. In J. Grefenstette, editor, *Proceedings of the 2nd International Conference on Genetic Algorithms*, pages 14–21, Cambridge, MA, 1987.

M.O. Ball. Computational complexity of network reliability analysis: An overview. *IEEE Transactions on Reliabilty*, 35(3):230–239, August 1986.

M.O. Ball, C. Colbourn, and J. Provan. Chapter 11: Network Reliability. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Network Models*, volume 7 of *Handbooks in Operations Research and Management Science*, pages 673–762. Elsevier, 1995.

N. Baumann and M. Skutella. Solving evacuation problems efficiently: Earliest arrival flows with multiple sources. *Mathematics of Operations Research*, 34(2):499–512, 2009.

R. Bechhofer, T. Santner, and D. Goldsman. *Design and Analysis of Experiments for Statisical Selection, Screening, and Multiple Comparisons*. John Wiley & Sons, 1995.

R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

G. Berlin. The use of directed routes for assessing escape potential. *Fire Safety Journal*, 14(2):126–135, May 1978.

D. Berry, A. Usmani, J. Torero, A. Tate, S. McLaughlin, S. Potter, A. Trew, R. Baxter, M. Bull, and M. Atkinson. Firegrid: Integrated emergency response and fire safety engineering for the future built environment. In *UK e-Science Programme All Hands Meeting (AHM-2005)*. UK e-Science Programme All Hands Meeting, September 2005.

L. Bianchi, M. Birattari, M. Chiarandini, M. Manfrin, M. Mastrolilli, L. Paquete, O. Rossi-Doria, and T. Schiavinotto. Metaheuristics for the Vehicle Routing Problem with Stochastic Demands. In X. Yao, E. Burke, J. Lozano, J. Smith, J. Merelo Guervós, J. Bullinaria, J. Rowe, P. Tiňo, A. Kabán, and H.-P. Schwefel, editors, *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature*, volume 3242, pages 450–460, 2004.

L. Bianchi, M. Birattari, M. Chiarandini, M. Manfrin, M. Mastrolilli, L. Paquete, O. Rossi-Doria, and T. Schiavinotto. Hybrid metaheuristics for the vehicle routing problem with stochastic demands. *Journal of Mathematical Modelling and Algorithms*, 5:91–110, 2006a.

L. Bianchi, M. Dorigo, L. Gambardella, and W. Gutjahr. Metaheuristics in Stochastic Combinatorial Optimization: a Survey. Technical Report 08 06, IDSIA - Dalle Molle Institute for Artificial Intelligence, March 2006b.

M. Birattari. *Tuning Metaheuristics: A Machine Learning Perspective*. Studies in Computational Intelligence. Springer Berlin / Heidelberg, 2009.

J. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer-Verlag, 1997.

C. Blum and A. Roli. Metaheuristics in Combinatorial Optimisation: Overview and Conceptual Comparison. *ACM Computing Surveys*, 35(3):268–308, September 2003.

J. Boesel. *Search and Selection for Large-Scale Stochastic Optimization*. PhD thesis, Northwestern University, 1999.

J. Boesel, B. Nelson, and S.-H. Kim. Using Ranking and Selection to "Clean Up" After Simulation Optimization. *Operations Research*, 51(5):814–825, 2003.

J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, 2002.

J. Branke and C. Schmidt. Selection in the Presence of Noise. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 766–777. Springer-Verlag, 2003.

J. Branke and C. Schmidt. Sequential Sampling in Noisy Environment. In *Parallel Problem Solving for Nature VIII*, pages 202–211. Springer-Verlag, 2004.

J. Branke, C. Schmidt, and H. Schmeck. Efficient Fitness Estimation in Noisy Environments. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 243–250. Morgan Kaufmann, 2001.

J. Branke, C. Schmidt, and S. Chick. New Developments in Ranking and Selection: An Empirical Comparison of the Three Main Approaches. In M. Kuhl, N. Steiger, F. Armstrong, and J. Joines, editors, *Proceeding of the 2005 Winter Simulation Conference*, pages 7660–777, 2005.

J. Branke, S.E. Chick, and C. Schmidt. Selecting a Selection Procedure. *Management Science*, 53(12):1916–1932, 2007.

P. Bratley, B. Fox, and L. Schrage. *A Guide to Simulation.* Springer-Verlag, 2nd edition, 1987.

P. Buchholz and A. Thümmler. Enhancing Evolutionary Algorithms with Statistical Selection Procedures for Simulation Optimization. In *Proceedings of the 2005 Winter Simulation Conference*, 2005.

R. Burkard, K. Dlaska, and B. Klinz. The Quickest Flow Problem. *Methods and Models of Operations Research*, 37:31–58, 1993.

X. Cai, D. Sha, and C. Wong. Time-Varying Universal Maximum Flow Problems. *Mathematical and Computer Modelling*, 33:407–430, 2001a.

X. Cai, D. Sha, and C. Wong. Time-varying minimum cost flow problems. *European Journal of Operational Research*, 131:352–374, 2001b.

P. Calégari, G. Coray, A. Hertz, D. Kobler, and P. Kuonen. A Taxonomy of Evolutionary Algorithms in Combinatorial Optimization. *Journal of Heuristics*, 5:148–158, 1999.

E. Cantú-Paz. Adaptive sampling for noisy problems. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 947–958, 2004.

M. Carey and C. Hendrickson. Bounds on expected performance of networks with links subject to failure. *Networks*, 14(3):439–456, 1984.

L. Chalmet, R. Francis, and P. Saunders. Network Models for Building Evacuation. *Management Science*, 28(1):86–105, January 1982.

C.-H. Chen. A lower bound for the correct subset-selection probability and its application to discrete-event system simulations. *IEEE International Conference on Automatic Control*, 41(8):1227–1231, 1996.

C.-H. Chen, S. Wu, and L. Dai. Ordinal Comparison of Heuristic Algorithms Using Stochastic Optimization. *IEEE Transations on Robotics and Automaton*, 15(1):44–56, 1999.

C.-H. Chen, J. Lin, E. Yücesan, and S. Chick. Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization. *Journal of Discrete Event Dynamic Systems*, 10(3):251–270, July 2000.

C.-H. Chen, E. Yücesan, L. Dai, and H.-C. Chen. Optimal budget allocation for discrete-event simulation experiments. *IIE Transactions*, 42:60–70, 2010.

H.-C. Chen, L. Dai, C.-H. Chen, and E. Yücesan. New Development of Optimal Computing Budget Allocation for Discrete Event Simulation. In K. Andradóttir, K. Healy, D. Withers, and B. Nelson, editors, *Proceedings of the 1997 Winter Simulation Conference*, pages 334–341, 1997.

L. Chen and E. Miller-Hooks. The Building Evacuation Problem with Shared Information. *Naval Research Logistics*, 55:363–376, 2008.

S. Chick and K. Inoue. New Two-Stage and Sequential Procedures for Selecting the Best Simulated System. *Operations Research*, 49(5):732–743, 2001.

S.E. Chick. Subjective Probability and Bayesian Methodology. In S. Henderson and B. Nelson, editors, *Handbooks in Operations Research and Management Science*, volume 17, chapter 9, pages 225–257. Elsevier B.V., 2006.

W. Choi, R. Francis, H. W. Hamacher, and S. Tufekci. Network Models of Building Evacuation Problems with Flow-Dependent Exit Capacities. In *Proceedings of the Tenth International Conference Operational Reasearch 1984*, 1984.

W. Choi, H. W. Hamacher, and S. Tufekci. Modeling of building evacuation problems by network flows with side constraints. *European Journal of Operational Research*, 35:98–110, 1988.

C. Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191:1245–1287, 2002.

L. Dai. Convergence Properties of Ordinal Comparison in the Simulation of Discrete Event Dynamic Systems. *Journal of Optimization Theory and Applications*, 91(2): 363–388, November 1996.

C. Davies and P. Lingras. Genetic Algorithms for Rerouting Shortest Paths in Dynamic and Stochastic Networks. *European Journal of Operational Research*, 144:27–38, 2003.

W. Davis. Intelligent Building Response. *Fire Protection Engineering*, 33(Winter): 26,30,32,34,36, 2007. URL `http://www.fpemag.com/archives/article.asp?issue_id=40&i=269`.

M. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, 1970.

A. Di Pietro, L. While, and L. Barone. Applying Evolutionary Algorithms to Problems with Noisy, Time-consuming Fitness Functions. In *Proceedings of the Congress on Evolutionary Computation*, pages 1254–1261, 2004.

F. Easton and N. Mansour. A distributed genetic algorithm for deterministic and stochastic labor scheduling problems. *European Journal of Operational Research*, 118:505–523, 1999.

G. Fishman. A Monte Carlo Sampling Plan for Estimating Network Reliability. *Operations Research*, 34(4):581–594, August 1986.

G. Fishman and T. Shaw. Evaluating Reliability of Stochastic Flow Networks. *Probability in the Engineering and Information Sciences*, 3:493–509, 1989.

J. Fitzpatrick and J. Grefenstette. Genetic Algorithms in Noisy Environments. *Machine Learning*, 3:101–120, 1988.

L. Fleischer. Faster Algoritms for the Quickest Transshipment Problem. In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 147–156, 1998.

L. Fleischer and É. Tardos. Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, 23(3-5):71–80, 1998.

L. Fogel, A. Owens, and M. Walsh. *Artificial intelligence through simulation evolution*. John Wiley & Sons, New York, USA, 1966.

L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, New Jersey, 1962.

O. Frank and W. Gaul. On Reliability in Stochastic Graphs. *Networks*, 12(2):119–126, 1982.

T. French, J. van Hemert, S. Potter, G. Wickler, and A. Tate. An Evolutionary Algorithm for the Safest Weighted Escape Problem in Stochastic, Time-Varying Networks. In *Metaheuristics International Conference*, Hamburg, Germany, July 2009.

M. Fu. Optimization for Simulation: Theory vs. Practice. *INFORMS Journal on Computing*, 14(3):192–215, September 2002.

T. Galili. Post hoc analysis for Friedman's Test (R code). `http://www.r-statistics.com/2010/02/post-hoc-analysis-for-friedmans-test-r-code/`, February 2010.

S. Gao and I. Chabini. Optimal routing policy problems in stochastic time-dependent networks. *Transportation Research Part B*, 40:93–122, 2006.

M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. Freeman, 1979.

M. Gen, R. Cheng, and D. Wang. Genetic Algorithms for Solving Shortest Path Problems. In *IEEE International Conference on Evolutionary Computation*, pages 401–406, Indianapolis, IN, April 1997.

M. Gen, R. Cheng, and S. Oren. Network design techniques using adapted genetic algorithms. *Advances in Engineering Software*, 32:731–744, 2001.

M. Gendreau and J-Y. Potvin. Metaheuristics in Combinatorial Optimization. *Annals of Operations Research*, 140:189–213, 2005.

M. Gendreau and J-Y. Potvin, editors. *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*. Springer, 2nd edition, 2010.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Reading, MA, 1989.

D. E. Goldberg, K. Deb, and J. Clark. Genetic Algorithms, Noise, and the Sizing of Populations. *Complex Systems*, 6:333–362, 1992.

G. Gopalakrishnan, B. Minsker, and D. Goldberg. Optimal Sampling in a Noisy Genetic Algorithm for Risk-Based Remediation Design. Master's thesis, University of Illinois at Urbana-Champaign, 2001.

W. Gutjahr, A. Hellmayr, and G. Pflug. Optimal Stochastic Single-Machine-Tardiness Scheduling by Stochastic Branch-and-Bound. *European Journal of Operational Research*, 117(2):396–413, 1999.

W. Gutjahr, C. Strauss, and E. Wagner. A Stochastic Branch-and-Bound Approach to Activity Crashing in Project Management. *INFORMS Journal on Computing*, 12 (2):125–135, 2000.

S. Gwynne, E. Galea, M. Owen, P. Lawrence, and L. Filippidis. A Review of the Methodologies Used in Evacuation Modelling. *Fire and Materials*, 23(6):383–388, August 1999.

R. Hall. The Fastest Path through a Network with Random Time-Dependent Travel Times. *Transportation Science*, 20:182–188, 1986.

H. W. Hamacher and S. A. Tjandra. Mathematical modeling of evacuation problems: A state of the art. In M. Schreckenberg and S. Sharma, editors, *Pedestrian and Evacuation Dynamics*, pages 227–266, 2002a.

H. W. Hamacher and S. A. Tjandra. Earliest arrival flow model with time dependent capacity for solving evacuation problems. In *Pedestrian and Evacuation Dynamics*, pages 267–276, 2002b.

H. W. Hamacher and S. Tufekci. On the use of lexicographic min cost flows in evacuation modeling. *Naval Research Logistics*, 34(4):487–503, 1987.

J. Hammersley and D. Handscomb. *Monte Carlo Methods*. Methuen & Co. Ltd., London, 2nd edition, 1964.

L. Han, S. Potter, G. Beckett, G. Pringle, S. Welch, S.-H. Koo, G. Wickler, A. Usmani, J. Torero, and A. Tate. FireGrid: An e-infrastructure for next-generation emergency response support. *Journal of Parallel and Distributed Computing*, 70:1128–1141, 2010.

H. Hedlund and M. Mollaghasemi. A Genetic Algorithm and an Indifference-Zone Ranking and Selection Framework for Simulation Optimization. In *Proceeding of the 2001 Winter Simulation Conference*, pages 417–421, 2001.

A. Hertz and D. Kobler. A framework for the description of evolutionary algorithms. *European Journal of Operational Research*, 126:1–12, 2000.

Y. Ho, R. Sreenivas, and P. Vakili. Ordinal Optimization of DEDS. *Journal of Discrete Event Dynamic Systems*, 2(2):61–88, 1992.

J. Holland. *Adaption in Natural and Artificial Systems*. The University of Michigan, Ann Harbor, MI, 1975.

M. Hollander and D. Wolfe. *Nonparametric Statistical Methods*. Wiley Series in Probability and Statistics. Wiley & Sons, 2nd edition, 1999.

S. Holm. A Simple Sequentially Rejective Multiple Test Procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

B. Hoppe and É. Tardos. Polynomial Time Algorithms for Some Evacuation Problems. In *5th Annual ACM–SIAM Symposium on Discrete Algorithms*, pages 433–441, 1994.

B. Hoppe and É. Tardos. The quickest transshipment problem. In *Proceedings of the 6th annual ACM-SIAM Symposium on Discrete Algorithms*, volume 25, pages 36–62, 1995.

J. Hsu. *Multiple Comparisons: Theory and Methods*. Chapman & Hall, 1996.

J. Inagaki, M. Haseyama, and H. Kitajima. A genetic algorithm for determining multiple routes and its applications. In *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems*, volume 6, pages 137–140, July 1999.

K. Inoue and S.E. Chick. Comparison of Bayesian and Frequentist Assessments of Uncertainty for Selecting the Best System. In *Proceedings of the 30th Conference on Winter Simulation*, pages 727–734, 1998.

K. Inoue, S. Chick, and C.-H. Chen. An Empirical Evaluation of Several Methods to Select the Best System. *ACM Transactions on Modelling and Computer Simulation*, 9(4):381–407, October 1999.

P. Jaillet. *Probabilistic Traveling Salesman Problems*. PhD thesis, Massachusetts Institute of Technology, 1985.

J. Jarvis and H. Ratliff. Some equivalent objectives for dynamic network flow problems. *Management Science*, 28(1):106–109, 1982.

O. Jellouli and E. Chatalet. Monte carlo simulation and genetic algorithm for optimising supply chain management in a stochastic environment. In *Proceedings of the 2001 IEEE Conference on Systems, Man, and Cybernetics*, volume 3, pages 1835–1839, 2001.

X. Ji. Models and algorithm for the stochastic shortest path problem. *Applied Mathematics and Computation*, 170:503–514, 2005.

Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, 2005.

Y. Jin and J. Branke. Evolutionary Optimization in Uncertain Environments - A Survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.

W. Jones and R. Bukowski. Critical Information for First Responders, Whenever and Wherever it is Needed. In *Proceedings of 9th International Interflam Conference*, volume 2, pages 1073–1082, Edinburgh, Scotland, September 2001. Interscience Communications Ltd.

W. Jones, D. Holmberg, W. Davis, D. Evans, S. Bushby, and K. Reed. Workshop to Define Information Needed by Emergency Responders during Building Emergencies. Technical Report NISTIR 7193, National Institute of Standards and Technology, Gaithersburg, MD, January 2005.

N. Kamiyama, N. Katoh, and A. Takizawa. An efficient algorithm for the evacuation problem in a certain class of networks with uniform path-lengths. *Discrete Applied Mathematics*, 157:3665–3677, 2009.

CJ Karbowicz and J. MacGregor Smith. A K-Shortest Paths Routing Heuristic for Stochastic Network Evacuation models. *Engineering Optimization*, 7(4):253–280, 1984.

S.-H. Kim and B. Nelson. Selecting the Best System. In S. Henderson and B. Nelson, editors, *Handbooks in Operations Research and Management Science*, volume 13, chapter 17. Elsevier B.V., 2006.

B. Kotnyek. An annotated overview of dynamic network flows. Technical Report 4936, INRIA, September 2003.

E. Kuligowski and R. Peacock. A Review of Building Evacuation Models. Technical report, National Institute of Standards and Technology, July 2005.

A. Law and W. Kelton. *Simulation Modelling and Analysis*. Industrial Engineering and Management Science. McGraw-Hill, 3rd edition, 2000.

P. Lin, S. Lo, H. Huang, and Yuen K. On the use of multi-stage time-varying quickest time approach for optimization of evacuation planning. *Fire Safety Journal*, 43: 282–290, 2008.

Y. Lin. A simple algorithm for reliability evaluation of a stochastic-flow network with node failure. *Computer & Operations Research*, 28:1277–1285, 2001.

Y. Lin. Study on the system capacity for a multicommodity stochastic-flow network with node failure. *Reliability Engineering & System Safety*, 78:57–62, 2002a.

Y. Lin. Using minimal cuts to evaluate the system reliability of a stochastic-flow network with failures at nodes and arcs. *Reliability Engineering & System Safety*, 75: 41–46, 2002b.

Y. Lin. Extend the quickest path problem to the system reliability evaluation for a stochastic-flow network. *Computer & Operations Research*, 30:567–575, 2003.

R. P. Loui. Optimal paths in graphs with stochastic and multidimensional weights. *Communications of the ACM*, 26:670–676, 1983.

G. Løvås. On performance measures for evacuation systems. *European Journal of Operational Research*, 85, 1995.

S. Lovetskii and I. Melamed. Dynamic Flows in Networks. *Automation and Remote Control*, 48:1417–1434, 1987.

K. Mak and Z. Guo. A Genetic Algorithm for Vehicle Routing Problems with Stochastic Demands and Soft Time Windows. In M. Jones, S. Patek, and B. Tawney, editors, *Proceedings of the 2004 Systems and Information Engineering Design Symposium*, pages 183–190, 2004.

Wai-Kei Mak, D. Morton, and R. Kevin Wood. Monte carlo bounding techniques for determining solution quality in stochastic programs. *Operations Research Letters*, 24:47–56, 1999.

A. Mandansky. Inequalities for Stochastic Linear Programming Problems. *Management Science*, 6(2):197–204, January 1960.

Z. Michalewicz. Constraint-Handling Techniques - Introduction. In T. Bäck, D. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter C5.1. IOP Publishing Ltd and Oxford University Press, 1997.

B. Miller. *Noise, Sampling and Efficient Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, May 1997.

B. Miller and D. Goldberg. Optimal Sampling for Genetic Algorithms. IlliGAL Report 96005, Univeristy of Illinois at Urbana-Champaign, August 1996.

E. Miller-Hooks. On-Line Information and Decision-Support in Building Egress. In *Proceedings of the 4th International Symposium on Human Behaviour in Fire*, pages 447–458, Cambridge, England, 2009.

E. Miller-Hooks and T. Krauthammer. An Intelligent Evacuation, Rescue and Recovery Concept. *Fire Technology*, 43(2):107–122, June 2007.

E. Miller-Hooks and H. Mahmassani. Least Possible Time Paths in Stochastic Time-Varying Networks. *Computer & Operations Research*, 25(12):1107–1125, 1998.

E. Miller-Hooks and S. Patterson. On Solving Quickest Time Problems in Time-Dependent, Dynamic Networks. *Mathematical Modelling and Algorithms*, 3:39–71, 2004.

E. Miller-Hooks and G. Sorrel. The Maximal Dynamic Expected Flows Problem for Emergency Evacuation Planning. In *TRB Research Board 87th Annual Meeting*, 2008.

P. Mirchandani. Shortest distance and reliability of probabilistic networks. *Computer & Operations Research*, 3:347–355, 1976.

M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1998.

A. Moraglio and R. Poli. Topological Interpretation of Crossover. In *Proceedings of Genetic and Evolutionary Computation Conference 2004*, pages 1377–1388, 2004.

A. Moraglio and R. Poli. Geometric Crossover for Sets, Multisets and Partitions. In *Parallel Problem Solving for Nature IX*, volume 4193, pages 1038–1047, 2006.

T. Munakata and D. Hashier. A Genetic Algorithm Applied to the Maximum Flow Problem. In *The Fifth International Conference on Genetic Algorithms*, pages 488–493, Urbana-Champaign, IL, July 1993.

H. Nagamochi and T. Ibaraki. Maximum Flows in Probabilistic Networks. *Networks*, 21(6):645–666, October 1991.

H. Nagamochi and T. Ibaraki. On Onaga's Upper Bound on the Mean Values of Probabilistic Maximum Flows. *IEEE Transactions on Reliabilty*, 41(2), June 1992.

M. Newman. *Handbook of Graphs and Networks*, chapter Random graphs as models of networks, pages 35–68. Wiley, 2003.

V. Norkin, G. Pflug, and A. Ruszczyński. A branch and bound method for stochastic global optimization. *Mathematical Programming*, 83:425–450, 1998.

K. Onaga. Bounds on the Average Terminal Capacity of Probabilistic Nets. *IEEE Transactions on Information Theory*, 14:766–768, 1968.

S. Opasanon. *On Finding Paths and Flows in Multicriteria, Stochastic and Time-Varying Networks*. PhD thesis, University of Maryland, 2004.

S. Opasanon and E. Miller-Hooks. The Safest Escape Problem. *The Journal of the Operational Research Society*, July 2008.

S. Opasanon and E. Miller-Hooks. Noisy Genetic Algorithm for Stochastic, Time-Varying Minimum Time Network Flow Problem. *Journal of the Transportation Research Record*, 2196:75–82, 2010.

A. Orda and R. Rom. Shortest-Path and Minimum-Delay Algorithms with Time-Dependent Edge-Length. *Association for Computing Machinery*, 37(3):607–625, July 1990.

A. Orda and R. Rom. Minimum Weights Paths in Time-Dependent Networks. *Networks*, 21(3):295–319, 1991.

A. Orda and R. Rom. On continuous network flows. *Operations Research Letters*, 17: 27–36, 1995.

C. Papadimitriou and K. Steiglitz. *Combinatorial Optimisation: Algorithms and Complexity*. Prentice-Hall, 1982.

A. Philpott. Continuous-Time Flows in Networks. *Mathematics of Operations Research*, 15(4):640–661, November 1990.

W. Powell, P. Jaillet, and A. Odoni. Chapter 3: Stochastic and Dynamic Networks and Routing. In M.O. Ball, T.L. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, pages 141–295. Elsevier, 1995.

D. Pretolani. A directed hypergraph model for random time dependent shortest paths. *European Journal of Operational Research*, 123(315-324), 2000.

M. Pullan. An Algorithm for a Class of Continuous Linear Programs. *SIAM Journal of Control and Optimization*, 31(6):1558–1577, November 1993.

M. Pullan. A Study of General Dynamic Network Programs with Arc Time-Delays. *SIAM Journal of Optimization*, 7(4):889–912, November 1997.

N. Radcliffe. Forma Analysis and Random Respectful Recombination. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 222–229, San Mateo, CA, 1991a. Morgan Kaufmann.

N. Radcliffe. Equivalence Class Analysis of Genetic Algorithms. *Complex Systems*, 5 (2):183–205, 1991b.

N. Radcliffe. Genetic Set Recombination and its Application to Neural Network Topology Optimisation. Technical Report EPCC-TR-91-21, Edinburgh Parallel Computing Centre, University of Edinburgh, 1991c.

N. Radcliffe. Genetic Set Recombination. In D. Whitley, editor, *Foundations of Genetic Algorithms 2*. Morgan Kaufmann, San Mateo, CA, 1992.

N. Radcliffe and F. George. A Study in Set Recombination. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, CA, 1993. Morgan Kaufmann.

S. Rana, D. Whitley, and R. Cogswell. Searching in the Presence of Noise. In *Parallel Problem Solving for Nature*, volume 1141, pages 198–207. Springer-Verlag, 1996.

R. Rechenberg. *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der biologischen Evolution*. Fromman-Holzboog, Stuttgart, Germany, 1973.

S. Ross. *Introduction to Probability Models*. Elsevier, 9th edition, 2007.

S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Inc, 2nd edition, 2003.

A. Sadek, B. Smith, and M. Demetsky. Dynamic traffic assignment: Genetic algorithms approach. *Transportation Research Record*, 1588:95–103, 1997.

Y. Sano and H. Kita. Optimization of Noisy Fitness Functions by Means of Genetic Algorithms Using History of Search. In *Parallel Problem Solving for Nature*, volume 1917, pages 571–580, 2000.

C. Schmidt. *Evolutionary Computation in Stochastic Environments*. PhD thesis, University of Karlsruhe, 2007.

C. Schmidt, J. Branke, and S. Chick. Integrating Techniques from Statistical Ranking into Evolutionary Algorithms. *Applications of Evolutionary Computing*, pages 752–763, 2006.

SFPE. *SFPE Handbook for Fire Protection Engineering*. National Fire Protection Association, 3rd edition, January 2002.

D. Singh, A. Ibrahim, T. Yohanna, and J. Sing. An Overview of the Applications of Multisets. *Novi Sad Journal of Mathematics*, 37(2):73–92, 2007.

J. Smalley. Risk-Based In Situ Bioremediation Design. Master's thesis, University of Illinois at Urbana-Champaign, 1998.

P. Stagge. Averaging Efficiently in the Presence of Noise. In *Parallel Problem Solving for Nature V*, volume 1498, pages 188–197, 1998.

J. Sudhir Ryan Daniel and C. Rajendran. A simulation-based genetic algorithm for inventory optimization in a serial supply chain. *International Transactions in Operational Research*, 12:101–127, 2005.

K. Talebi and J. MacGregor Smith. Stochastic network evacuation models. *Computer & Operations Research*, 12(6):559–577, 1985.

S. A. Tjandra. *Dynamic Network Optimization with Application to the Evacuation Problem*. PhD thesis, University of Kaiserslautern, May 2003.

R. Upadhyay, G. Pringle, G. Beckett, S. Potter, L. Han, S. Welch, A. Usmani, and J. Torero. An architecture for an integrated fire emergency response system for the built environment. In *9th Symposium of the International Association for Fire Safety Science*, 2008.

L. Valiant. The Complexity of Enumeration and Reliability Problems. *SIAM Journal of Computing*, 8(3):410–421, 1979.

H. Varia and S. Dhingra. Dynamic optimal traffic assignment and signal time optimization using genetic algorithms. *Computer-Aided Civil and Infrastructure Engineering*, 19:260–273, 2004.

J. Watson, S. Rana, D. Whitley, and A. Howe. The Impact of Approximate Evaluation on the Performance of Search Algorithms for Warehouse Scheduling. *Journal of Scheduling*, 2(2):79–98, 1999.

B. Welch. The Significance of the Difference Between Two Means when the Population Variances are Unequal. *Biometrika*, 29(3/4):350–362, February 1938.

D. Whitley. The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. In J. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*. Morgan Kaufman, 1989.

M. Yokoyama and H. Lewis III. Optimization of the stochastic dynamic production cycling problem by a genetic algorithm. *Computer & Operations Research*, 30: 1831–1849, 2003.

Y. Yoshitomi. A genetic algorithm approach to solving stochastic job-shop scheduling problems. *International Transactions in Operational Research*, 9:479–495, 2002.

Y. Yoshitomi, H. Ikenoue, T. Takeba, and S. Tomita. Genetic algorithm in uncertain environments for solving stochastic programming problem. *Journal of the Operations Research Society of Japan*, 43(2):266–290, 2000.