



2059713

Abstract

In real world applications, the path-planning problem is one that is multi-criteria in nature though given the complexity of the task is one that is often condensed to a single criterion issue, either by the consideration of only a single objective or condensing several criteria into a single metric through aggregation or weighting. The thesis describes research that has led to the development and application of heuristic techniques in order to optimise shortest paths where more than one single criterion is to be evaluated. The techniques are described and demonstrated, and their effectiveness established by testing them using synthetic and real world datasets. The scalability of the heuristics to increasing numbers of criteria is demonstrated.

Heuristic techniques are able to solve the Multi Objective Shortest Path Problem (MSPP). In several cases, the performance of the techniques outperform traditional algorithmic methods by over 30-50% in terms of runtime, whilst returning a good approximation of the optimal set of paths. Promising alternative methods for candidate path generation are presented. These offer a much faster runtime for the evolutionary algorithm approach, which is able to complete a run on larger graphs in around five seconds. Further, several potentially more promising methods have been identified for future work, these would lead to increased performance of the mechanisms with a decreased runtime whilst returning a more complete set of optimal solutions.

Metaheuristics for Multi Criteria Path Optimisation

Andrew Olden BSc. MSc.



Thesis Submitted in Partial Fulfilment of the Requirements for

Candidature for the Degree of Doctor of Philosophy

Department of Computing and Mathematical Sciences, Faculty of Advanced Technology

University of Glamorgan

September 2012

“Callum was a bit surprised that Rafik didn’t share in the joys of the beauties of applied linear programming, but then it took all sorts didn’t it?”

Anne McCaffrey & Margaret Ball, Acorna: The Unicorn Girl

Statement of Originality

This is to certify that, except where specific reference is made, the work described in this thesis is the result of the candidate's research. Neither this thesis, nor any part of it, has been presented, or is currently submitted, in candidature for any degree at any other University.

Signed (Candidate)

Signed (Director of Studies)

Date

Acknowledgments

I would like to take the opportunity to thank my supervision team during the course of this work. Professors George Taylor and Stuart Cole, and Mr. Colin Morris. Particular thanks are due to Colin who did not let it (perhaps it was something I said) at the first opportunity. The man is a martyr, especially after having to put up with my attempts to convert 41293 from decimal to hexadecimal in Level 1 Systems Architecture.

I would also like to thank (no particular order), Professors Gary Higgs and Andrew Ware, Dr Rob Berry, Dr Karen Fitzgibbon, Dr Jim Moon, Alice Lau, Owen Clark and Steve Woodward for proof reading this work and attempting to make sense of my very poor spelling and grammar. Little Chriss' (Chris Wells) annual encouragement at Easter in Nerja was also gratefully received, as was his advice on the fine art of avoiding oversized plant pots and for the opportunity to partake in day trips to exotic locations. I would also thank Dr Christine Mumford of Cardiff University, whose feedback and criticisms (all constructive and beneficial) proved invaluable.

On a personal level, I would like to thank Jack, MJ, Trevor, Big Draig and the FRG for their support during the past few (hah!) years. Oh, and Mr B. Nearly forgot that there. His threats of marathon 'Lost' bank holiday weekends certainly proved inspirational towards the end of what has proven to be an exceptionally long journey. *Thank you.*

PS. Its A41C. I think.

Abstract

In real world applications, the path-planning problem is one that is multi-criteria in nature though given the complexity of the task is one that is often condensed to a single criterion issue, either by the consideration of only a single objective or condensing several criteria into a single metric through aggregation or weighting. The thesis describes research that has led to the development and application of heuristic techniques in order to optimise shortest paths where more than one single criterion is to be evaluated. The techniques are described and demonstrated, and their effectiveness established by testing them using synthetic and real world datasets. The scalability of the heuristics to increasing numbers of criteria is demonstrated.

Heuristic techniques are able to solve the Multi Objective Shortest Path Problem (MSPP). In several cases, the performance of the techniques outperform traditional algorithmic methods by over 30-50% in terms of runtime, whilst returning a good approximation of the optimal set of paths. Promising alternative methods for candidate path generation are presented. These offer a much faster runtime for the evolutionary algorithm approach, which is able to complete a run on larger graphs in around five seconds. Further, several potentially more promising methods have been identified for future work, these would lead to increased performance of the mechanisms with a decreased runtime whilst returning a more complete set of optimal solutions.

List of Equations

Equation 1.1 Simple Weighting	5
Equation 2.1 The Path Description Vector	23
Equation 2.2 The Multi Objective Shortest Path Problem (MSPP)	24
Equation 2.3 Barabasi and Albert Probability Value	32
Equation 2.4 A* Heuristic Mechanism	46
Equation 2.5 Distance Calculation	47
Equation 2.6 ALT* Heuristic Equation	52
Equation 2.7 De Meo Centrality Metric	69
Equation 2.8 Agarwal Probability Metric	72
Equation 3.1 Fonseca and Fleming Ranking Equation	111
Equation 3.2 Boltzmann Distribution	124
Equation 3.3 ONVGR Metric	139
Equation 3.4 Error Ratio Metric	140
Equation 3.5 Generation Distance Metric	141
Equation 3.6 Spacing Metric Metric (Ripon, 2007)	143
Equation 4.1 Absolute Delta Calculation	170

List of Abbreviations

ADV	Absolute Distance Vector
AI	Artificial Intelligence
BFS	Breadth First Search
BH	Binary Heap
BSP	Bi-criterion Shortest Path Problem
CPU	Central Processing Unit
DM	Decision Maker
EA	Evolutionary Algorithm
ER	Error Ratio
EU	European Union
FH	Fibonacci Heap
FIFO	First In First Out
GA	Genetic Algorithm
GD	Generational Distance
GIS	Geographical Information System
GNVG	Generational Non-dominated Vector Generation
ID	Identification
ITN	Integrated Transport Network
LIFO	Last In First Out
MFE	Maximum From Error (Also MPFE)
MOEA	Multi Objective Evolutionary Algorithm
MOGA	Multi Objective Genetic Algorithm
MOO	Multi Objective Optimisation
MOP	Multi Objective Problem
MOTS	Multi Objective Tabu Search
MNTS	Multi Nominal Tabu Search
MPFE	Maximum Pareto Front Error (Also MFE)
MSPA	Multi Objective Shortest Path Algorithm
MSPP	Multi Objective Shortest Path Problem
NAMOA*	New Multi Objective A*
NE	Newcastle Emlyn
NPGA	Niched Pareto Genetic Algorithm
NSGA	Non-Dominated Sorting Genetic Algorithm
ONGVR	Overall Non Generated Vector Ratio

OS	Ordnance Survey
PAES	Pareto Archiving Evolutionary System
PDV	Path Description Vector
PO	Pareto Optimal
RW	Random Walk
SA	Simulated Annealing
SP	Shortest Path
SPEA	Strength Pareto Evolutionary Algorithm
TS	Tabu Search
UK	United Kingdom
UTRP	Urban Transit Routing Problem
VEGA	Vector Evaluated Genetic Algorithm

List of Tables

Table 1.1 Shortest Path Across Single Criteria Graph.....	6
Table 1.2 Shortest Path Across Total Sum Graph.....	6
Table 1.3 Shortest Path Across Weighted Sum Graph	7
Table 2.1 Applications of Graphs (Ahuja et al, 1993)	19
Table 2.2 Heap Operation Times (Cormen et al, 2001).....	43
Table 2.3 Comparison of Hierarchy Based Approaches	56
Table 2.4 Runtimes Seen in the Delling et al (2011) Approach	63
Table 2.5 Sizes of Various Social Networks.....	67
Table 2.6 Criteria Costs of Solutions Provided in Figure 2.28	76
Table 2.7 All Paths Between Vertices A and D on Example Graph.....	79
Table 2.8 Pareto Table with Paths Through Vertex E Removed	80
Table 2.9 Pareto Optimal Paths.....	81
Table 2.10 NAMOA* Real World Test Graphs	94
Table 3.1 Comparison of Sets $PfTRUE$ and $PfAPPROX$ Based on Figure 3.10.....	138
Table 3.2 Error Ratio (ER) Table.....	140
Table 3.3 Generational Distance Measures.....	142
Table 4.1 Random Graph Sizes.....	153
Table 4.2 SPRAND Attribute Types.....	157
Table 4.3 Road Types Available via the Mastermap ITN Layer	161
Table 4.4 Newcastle Emlyn Statistics.....	161
Table 4.5 Cardiff Statistics.....	163
Table 4.6 London Statistics.....	164
Table 4.7 Feature and Vertex Ratios on Urban and Rural Roads	169
Table 5.1 Runtime of S&A Algorithm on Real World Graphs	214
Table 5.2 Runtime of S&A Algorithm on Random Graphs	214
Table 5.3 Performance of Climaco and Martins Algorithm with Low Delta Values ...	218
Table 5.4 Performance of Climaco and Martins Algorithm with High Delta Values...	218
Table 5.5 Comparison of Results Seen in Climaco and Martins Approach	219
Table 5.6 Total Paths Generated Per Run of Genetic Algorithm.....	226

Table 5.7 Analysis of Runtime Differentials	229
Table 5.8 Comparative Qualities of Distance Metrics	235
Table 5.9 Runtime of Various MSPP Techniques on Random Graphs	240
Table 6.1 Number of Unique Paths Generated using Random Walk	255
Table 6.2 Parameter Sets Used in GA Analysis.....	268
Table 6.3 Simulated Annealing Parameters	278
Table 6.4 <i>PfTRUE</i> Counts Against Acquired Using Simulated Annealing	282
Table 7.1 National Level Graph Sizes	296
Table A.1 Simple Example of Erdos-Reyni Random Graph (n=5, m=8).....	A-339
Table A.2 Example of Gilbert Graph with p=0.1	A-339
Table A.3 Example of Gilbert Graph with p=0.6	A-339
Table A.4 Example of Gilbert Graph with p=0.8	A-339
Table A.5 Watts and Strogatz Example Graph (p=0.0, k = 4, n=10).....	A-340
Table A.6 Watts and Strogatz example graph (p=0.5, k = 4, n=10)	A-340
Table A.7 Watts and Strogatz Example Graph (p=0.8, k = 4, n=10).....	A-341
Table A.8 Barabasi and Albert Initial Random Graph.....	A-341
Table A.9 Barabasi and Albert Complete Random Graph.....	A-342
Table B.1 Runtime of Heuristics on Random Graphs.....	B-343
Table B.2 Runtime of Heuristics on Real World Graphs.....	B-343
Table B.3 Summary of Results using GA (Random Walk Based)	B-344
Table B.4 Summary of Results using GA (K Geodesic Based).....	B-345
Table B.5 Summary of Quality Tests using PAES	B-346
Table B.6 Summary of Quality Tests Using Tabu Search	B-347
Table B.7 Summary of Quality Tests Using Simulated Annealing	B-348
Table B.8 Presence of Local Optimal Paths for Each Algorithm	B-349
Table B.9 Average Distance of Locally Optimal Solutions from <i>PfTRUE</i>	B-350

List of Figures

Figure 1.1 Simple Example of a Single Criteria Graph	5
Figure 1.2 Simple Example of a Multi Criteria Graph.....	6
Figure 2.1 Various Graph Types (Sedgewicke, 2003).....	20
Figure 2.2 Simple Graph Without Costs.....	21
Figure 2.3 Simple Directed Graph	21
Figure 2.4 Simple Directed Graph with Costs	22
Figure 2.5 Example Path Description Vector (PDV).....	23
Figure 2.6 Centroid of a Rectangle	25
Figure 2.7 Example Graph Generated using Erdos-Reyni Model	27
Figure 2.8 Example Graph Generated using the Gilbert Model ($p=0.1$)	29
Figure 2.9 Example Graph Generated using the Gilbert Model ($p=0.6$)	29
Figure 2.10 Example Graph Generated using the Gilbert Model ($p=0.8$)	30
Figure 2.11 Example Graph Generated using the Watts and Strogatz Model ($p=0.1$) ...	31
Figure 2.12 Example Graph Generated using the Watts and Strogatz Model ($p=0.5$) ...	31
Figure 2.13 Example Graph Generated using the Watts and Strogatz Model ($p=0.9$) ...	31
Figure 2.14 Example Graph Generated using the Barabasi and Albert Model ($m_0=4$)..	32
Figure 2.15 Example Graph Generated using the Barabasi and Albert Model ($v=10$)...	33
Figure 2.16 Example Network with Prescribed Degree Sequence	35
Figure 2.17 Papers on Shortest Path Analysis Published Between 2000-2012	38
Figure 2.18 Basic Binary Heap Structure (Cormen et al, 2001)	42
Figure 2.19 Basic Fibonacci Heap Structure (Ahuja et al, 1993)	42
Figure 2.20 Organisation of Heap in Figure 2.19	42
Figure 2.21 Single Bucket Structure (Dial, 1979).....	43
Figure 2.22 Runtime of Shortest Path Algorithms (Zhan, 1997).....	45
Figure 2.23 ALT* Triangulation Graph.....	52
Figure 2.24 Simple Test Graph for Hierarchies	53
Figure 2.25 Simple Test Graph from Figure 2.24 After Shortcut Mechanism	53
Figure 2.26 Example grid model of Bast et al (2006).....	59
Figure 2.27 Example Pareto Optimal Front and Space.....	75
Figure 2.28 Example of 2D Search Space with Four Solutions.....	76

Figure 2.29 Example Two Criteria Graph.....	78
Figure 2.30 Pareto Search Space of Table 2.3	80
Figure 2.31 Filtered Pareto Search Space	81
Figure 2.32 Pareto Optimal Solutions for Example Graph	81
Figure 2.33 Pareto Optimal Paths Through Graph.....	82
Figure 2.34 Individual Unique Paths Through Example Graph	83
Figure 2.35 Publication on Multi Objective Path Analysis Since 2000.....	86
Figure 2.36 The Skriver Taxonomy of MSPP Techniques	89
Figure 2.37 Initial Search Results (Coutinho-Rodrigues et al. (1994)).....	92
Figure 2.38 Refined Search Results (Coutinho-Rodrigues et al, 1994).....	92
Figure 3.1 Flow of the Basic Genetic Algorithm (Hoitkotter and Beasley, 2000)	105
Figure 3.2 Outline of General Crossover Procedure	108
Figure 3.3 Outline of Mutation Operator	108
Figure 3.4 Outline of VEGA Algorithm	110
Figure 3.5 Visualisation of the Goldberg Ranking Methodology.....	112
Figure 3.6 Crowd Based Nicheing Example.....	113
Figure 3.7 Multi Objective Processing Types.....	116
Figure 3.8 Example Energy Measure Determination of Two Solutions.....	130
Figure 3.9 Multi Level Memory Model (Jaeggi et al, 2008)	132
Figure 3.10 Example of <i>PfTRUE</i> and <i>PfAPPROX</i>	138
Figure 3.11 Hyper Volume Rectangles.....	144
Figure 3.12 Hyper Volume Front.....	144
Figure 4.1 UML Diagram Representing the Graph Structure.....	149
Figure 4.2 Visualisation of Simple SPRAND graph.....	158
Figure 4.3 Single Criteria Graph from SPRAND	158
Figure 4.4 Single Criteria Graph Following Multi Criteria Conversion	158
Figure 4.5 General Overview of Newcastle Emlyn (Crown Copyright, 2011)	162
Figure 4.6 General Overview of Cardiff Area (Crown Copyright, 2011)	163
Figure 4.7 General Overview of London Area (Crown Copyright, 2011).....	165
Figure 4.8 Edges in Selected Real World Graphs.....	166
Figure 4.9 Vertices in Selected Real World Graphs	166
Figure 4.10 Almost Straight-Line Path Requiring Few Links	168

Figure 4.11 Road Feature with High Geodesic Distance	168
Figure 4.12 Overview of Translation Software.....	173
Figure 4.13 Parameter Setting for Translation Software	174
Figure 4.14 Coordinate Information for Cardiff250 Subset	177
Figure 4.15 Graph Information for Cardiff250 Subset	177
Figure 4.16 Simple Example Graph.....	179
Figure 4.17 Highlighted Path From Figure 4.16.....	179
Figure 4.18 Binary String Path Representation	180
Figure 4.19 Example Graph with ‘Dead Ends’ Highlighted.....	184
Figure 4.20 Simple Graph for Mutation Operation.....	189
Figure 4.21 Selected Parent Paths Between Vertices 1 and 4.....	190
Figure 4.22 Results of Crossover Procedure.....	190
Figure 4.23 Example Graph for the Mutation Operator.....	191
Figure 4.24 Input PDV Into the Mutation Operator.....	192
Figure 4.25 Output PDV From the Mutation Operator.....	192
Figure 5.1 Graph of Multi Criteria Dijkstra Approach Runtimes.....	212
Figure 5.2 Runtime of S&A Algorithm on Real World Graphs	214
Figure 5.3 Runtime of S&A on Random Graphs.....	215
Figure 5.4 Comparison of the Paths Required for C&M Approach	219
Figure 5.5 Comparison of the Runtime Required for C&M Approach	220
Figure 5.6 Runtime of Heuristics on Random Graphs.....	222
Figure 5.7 Runtime of Heuristics on Real World Graphs.....	224
Figure 5.8 Runtime of the Genetic Algorithm for Various Numbers of Criteria.....	228
Figure 5.9 Tests Where $D(PfAPPROX, PfTRUE) \leq 0$	230
Figure 5.10 Tests Where $D(PfAPPROX, PfTRUE) = 1$	231
Figure 5.11 Tests Where $D(PfAPPROX, PfTRUE) \geq 2$	231
Figure 5.12 Tests Producing High Quality Results.....	233
Figure 5.13 Number of Locally Optimal Solutions Produced by Heuristics	234
Figure 5.14 Average Distance of Locally Optimal Solutions to $PfTRUE$	235
Figure 5.15 Real World Example of $ PfTRUE \leq Approx $	236
Figure 5.16 Diverging Properties of MSPP Approaches	237
Figure 5.17 Runtime of Approaches on Small Sized Graphs	238

Figure 5.18 Runtime of approaches on a Medium sized graph.....	239
Figure 5.19 Runtime of Approaches on a Large Graph	239
Figure 5.20 Highlighting Figure 5.16 with Multi Objective Dijkstra Approach	241
Figure 5.21 Run Approach on Small Graphs	242
Figure 5.22 Run Model on Medium Sized Graphs	243
Figure 5.23 Comparison of Heuristics on Random Graphs (Run Time)	244
Figure 5.24 Comparison of Heuristic on Random Graphs (Quality)	244
Figure 5.25 Run Model on Large Graphs	246
Figure 6.1 Graph Admission Rates Achieved by Random Walking	253
Figure 6.2 Vertex Admission During a Run of the Genetic Algorithm	254
Figure 6.3 Unique Paths Discovered by Random Walk	254
Figure 6.4 Length of Optimal Paths on Road Networks.....	256
Figure 6.5 Path Lengths Obtained using K-Geodesics	257
Figure 6.6 Path Lengths Obtained using Random Walking.....	258
Figure 6.7 Summary of Quality Tests on Genetic Algorithm (Random Walking).....	261
Figure 6.8 Summary of Quality Tests on Genetic Algorithm (K-Geodesic)	262
Figure 6.9 Summary of Quality Tests on the Tabu Search	263
Figure 6.10 Summary of Quality Tests on Simulated Annealing	265
Figure 6.11 Summary of Quality Tests on PAES	266
Figure 6.12 Runtime of Random Walking Based GA	269
Figure 6.13 Runtime of K-Geodesic Based GA.....	270
Figure 6.14 ONVGR on 2D Graphs Using the GA (Random Walking)	270
Figure 6.15 ONVGR on 3D Graphs Using the GA (Random Walking)	271
Figure 6.16 ONVGR on 2D Graphs Using the Tabu Search	274
Figure 6.17 Effect of Increasing Tabu List Size on ONVGR (Set D)	274
Figure 6.18 Example of <i>PfAPPROX</i> Containing Non-Optimal Solutions.....	275
Figure 6.19 Example Fronts <i>PfTRUE</i> & <i>PfAPPROX</i> Obtained using Tabu Search	276
Figure 6.20 Comparison of <i>PfTRUE</i> & <i>PfAPPROX</i> With Locally Optimal Solutions.....	277
Figure 6.21 ONVGR on 2D graphs Using Simulated Annealing	279
Figure 6.22 Comparison of <i>PfTRUE</i> and <i>PfAPPROX</i> From Simulated Annealing.....	281
Figure 6.23 Size of <i>PfAPPROX</i> Over SA Processing Run	281
Figure 6.24 ONVGR on 2D Graphs Using PAES	283

Figure 7.1 View of a Simple Uncondensed Road Segment	298
Figure 7.2 View of a Simple Condensed Road Segment	298
Figure C.1 UML Diagram For Graph Structure.....	C-351
Figure C.2 UML Diagram for Domination System	C-352
Figure C.3 UML Diagram for Random Walk System	C-353
Figure C.4 UML Diagram for Example Algorithm (Tabu Search)	C-354

List of Algorithms

Algorithm 2.1 Erdos-Reyni Random Graph Generation Model	27
Algorithm 2.2 Gilbert Random Graph Generation Model	28
Algorithm 2.3 Barabasi and Albert Random Graph Generation Model	34
Algorithm 2.4 SPRAND Random Graph Generator Model	37
Algorithm 2.5 Dijkstra's Shortest Path Algorithm	41
Algorithm 2.6 Simplified Contraction Hierarchy Construction.....	55
Algorithm 2.7 Calculation of Shortest Path Using the Bast et al (2006) Model.....	60
Algorithm 2.8 Pareto Front Extraction Algorithm.....	77
Algorithm 3.1 Goldberg Ranking Method.....	111
Algorithm 3.2 NPGA Outline	114
Algorithm 3.3 NPGA Specialised Binary Tournament Selection.....	115
Algorithm 3.4 Outline of the PAES Algorithm	118
Algorithm 3.5 PAES Test Archive Procedure	119
Algorithm 3.6 SPEA Algorithm.....	121
Algorithm 3.7 SPEA2 Algorithm.....	122
Algorithm 3.8 Basic Outline of Simulated Annealing.....	124
Algorithm 3.9 Simulated Annealing with Multiple Neighbours.....	126
Algorithm 3.10 Single Criteria Tabu Search.....	133
Algorithm 4.1 Examples of Graph Loading Procedures	151
Algorithm 4.2 Geodesic Calculation Algorithm (Newman, 2001)	156
Algorithm 4.3 Conversion to Multiple Criteria Algorithm	159
Algorithm 4.4 Extraction of PfTRUE from Real Road Graphs	172
Algorithm 4.5 Real Road Extraction Algorithm	178
Algorithm 4.6 Random Walk Algorithm	181
Algorithm 4.7 Remove Cycles Algorithm	182
Algorithm 4.8 Basic Outline of GA Approach	185
Algorithm 4.9 Genetic Operators Outline.....	187
Algorithm 4.10 Outline of Crossover Procedure	189
Algorithm 4.11 Outline of the PAES Algorithm	194

Algorithm 4.12 Outline of the PAES Archiving Strategy	195
Algorithm 4.13 PAES Update Grid Algorithm.....	196
Algorithm 4.14 PAES Find Location Algorithm	197
Algorithm 4.15 1+1 Mutation Operator	198
Algorithm 4.16 Outline of the Tabu Search.....	200
Algorithm 4.17 Distance Calculation for Indifferent Solutions.....	201
Algorithm 4.18 Probability of Acceptance Algorithm.....	202
Algorithm 4.19 Outline of Simulated Annealing Algorithm	203
Algorithm 4.20 Dominance Check Algorithm.....	204
Algorithm 4.21 Extraction of Pareto Optimal Front	204
Algorithm 4.22 Path Equality Algorithm.....	206
Algorithm 4.23 Population Contains Algorithm	207
Algorithm 4.24 Simple Population Domination Check	207
Algorithm 4.25 Calculation of the Path Description Vector.....	208
Algorithm 5.1 Application of Both SaA and CaM Approaches	247
Algorithm 5.2 Run Model of MSPP Approaches	248

Glossary

Annealing Schedule	Rate Of Decline Used In Simulated Annealing Algorithm
Edge	A Link Between Two Vertices
Generations	Number Of Iterations A Genetic Algorithm Runs
Graph	Set Comprising $G = (V , E)$
Vertex	A Minimal Element – End Point Of An Edge
Path	A Series Of Edges Linking A Source And Destination
PfAPPROX	Set Of Optimal Solutions Obtained From Algorithms
PfTRUE	Set Of Complete Optimal Solutions
Population	Number Of Paths Consider Per GA Iteration
Tabu List	Current List Of Inadmissible Move
Temperature	Function Of Annealing Schedule, Effecting Probability Of Acceptance Of Inferior Solutions

Contents

Statement of Originality	i
Acknowledgments	ii
Abstract.....	iii
List of Equations	iv
List of Abbreviations	v
List of Tables	vii
List of Figures.....	ix
List of Algorithms	xiv
Glossary.....	xvi
Contents	xvii
1. Introduction	2
1.1. Background	2
1.2. The Path Planning Problem	4
1.3. Applications of Path Planning	7
1.3.1. Transportation Networks.....	7
1.3.2. Computer Networks	8
1.3.3. Fighting Organized Crime.....	8
1.3.4. Medical Applications	9
1.4. Decision Making in Path Planning	9
1.5. Research Definition	13
1.6. Research Aims.....	14
1.7. Thesis Format	15

2. Graph Theory and Shortest Path Analysis	18
2.1. Background	18
2.2. Terminology	20
2.3. Random Graph Generation.....	25
2.3.1. The Erdos and Renyi Model	26
2.3.2. The Gilbert Model.....	28
2.3.3. The Barabási–Albert Scale Free Model	30
2.3.4. Alternative Methods of Random Graph Generation	33
2.4. Single Criteria Path Optimisation.....	37
2.4.1. Alternative Shortest Path Algorithms	46
2.4.2. K Shortest Paths	48
2.5. State of the Art in Single Criteria Shortest Path Analysis	50
2.5.1.1 Bi-directional Searching.....	51
2.5.1.2 Hierarchical Methods	52
2.5.1.3 Transit Node Approach	57
2.5.1.4 Multi Level Pre-processing	61
2.5.1.5 Practical Implementations	63
2.5.1.6 Summary of the State of the Art in Shortest Path Analysis	65
2.6. Social Network Analysis	65
2.7. Multi Criteria Path Optimisation.....	73
2.7.1. Worked Example of the MSPP	78
2.7.2. Existing Work on the MSPP	84
2.7.2.1 Heuristic and Evolutionary Approaches to the MSPP	95
2.8. Chapter Summary.....	101
3. Multi Objective Optimisation	104
3.1. Genetic Algorithms and Evolutionary Computation.....	104

3.1.1.	Non-Pareto Approaches	109
3.1.2.	Population Based Approaches.....	110
3.1.3.	Elitist Based Methodologies	115
3.2.	Multi Objective Simulated Annealing.....	123
3.3.	Multi Objective Tabu Search	131
3.4.	Quality Measurement in Multi Criteria Based Optimisation	136
3.4.1.	Convergence Measuring Metrics	138
3.4.2.	Metrics Coverage Distance, Coverage and Spread	141
3.5.	Chapter Summary	145
4.	Experimental Design	147
4.1.	Graph Data Structures	148
4.2.	Data Selection.....	152
4.2.1.	Randomly Generated Graphs	152
4.2.1.1	Calculation of Front PfTRUE.....	154
4.2.2.	Graph Format	155
4.2.3.	Real World Graphs.....	159
4.2.3.1	Newcastle Emlyn.....	161
4.2.3.2	Cardiff	162
4.2.3.3	London.....	164
4.2.3.4	Real World Data Observations.....	165
4.2.3.5	Calculation of Front PfTRUE.....	167
4.2.3.6	Data Extraction Process.....	171
4.3.	Path representation	179
4.4.	Random Path Generation.....	180
4.5.	A Genetic Algorithm for the MSPP	183
4.5.1.	Evolutionary Operators	186

4.5.1.1	Selection Operation	186
4.5.1.2	Crossover	188
4.5.1.3	Mutation	191
4.6.	A PAES Based Approach to the MSPP	192
4.7.	Tabu Search Algorithm	198
4.8.	Simulated Annealing Algorithm	201
4.9.	Dominance and extraction of <i>PfAPPROX</i>	204
4.10.	Support Algorithms	205
4.11.	Chapter Summary	208
5.	Runtimes and Scalability	211
5.1.	Algorithmic Solutions to the MSPP	211
5.1.1.	Dijkstra Algorithm Using Multiple Criteria	211
5.1.2.	Skriver and Anderson's Algorithm	213
5.1.3.	Climaco and Martins' Algorithm	215
5.1.4.	Observation Regarding the Algorithmic Approach	220
5.2.	Analysis of Runtimes of Heuristic Approaches	222
5.3.	Reviewing the Runtime Discrepancy	225
5.4.	Scalability of Criteria	227
5.5.	Summary of Quality Tests	229
5.6.	Admission of Locally Optimal Paths	233
5.7.	Cross Algorithm Analysis	236
5.7.1.	Summary Of Cross Algorithm Analysis	246
5.8.	Chapter Summary	248
6.	Behaviour Analysis	251
6.1.	Random Walking	251
6.2.	Summary of Quality Tests on Heuristic Approaches	258

6.2.1.	Summary of Quality Tests on the Genetic Algorithm	259
6.2.2.	Summary of Quality Tests on the Tabu Search	262
6.2.3.	Summary of Quality Tests on Simulated Annealing.....	264
6.2.4.	Summary of quality tests on the PAES	265
6.2.5.	Observations Gathered from the Quality Tests	266
6.3.	Analysis of Algorithm Operation	268
6.3.1.	Analysis of the Genetic Algorithm Operation	268
6.3.1.1	Observations of the Genetic Algorithm Operation.....	272
6.3.2.	Analysis of the Tabu Search Approach.....	273
6.3.2.1	Observations of the Tabu Search Operation.....	275
6.3.3.	Analysis of the Simulated Annealing Algorithm	278
6.3.3.1	Observations of the Simulated Annealing Operation.....	279
6.3.4.	Analysis of the PAES Algorithm	282
6.3.5.	Observations Regarding The Algorithms.....	283
6.4.	Limitations in Experiments Undertaken.....	284
6.5.	Chapter Summary	286
7.	Conclusions and Future Work.....	288
7.1.	Research Methodology	288
7.2.	Summary of Research Outcomes	289
7.2.1.	Research Aim 1	290
7.2.2.	Research Aim 2	291
7.2.3.	Research Aim 3	293
7.3.	General Observations	295
7.3.1.	Observations Regarding the use of ‘Memory’	295
7.3.2.	Scalability.....	295
7.3.3.	Development of <i>PfTRUE</i> on Real World Graphs	297

7.4. Future Work	297
7.4.1. Graph Generalisation	297
7.4.2. Analysis of Selection Methods	299
7.4.3. Hybrid Algorithms	299
7.5. Closing Comments	300
References	302
Appendix A: Example Random Graphs	A-339
Appendix B: Result Tables.....	B-343
Appendix C: UML Diagrams	C-351

Chapter One: Introduction

1. Introduction

The following chapter introduces the background to the research undertaken, namely the application of meta-heuristic algorithms to the multi objective shortest path problem (MSPP). An overview of how and why people often select what are perceived to be sub optimal routes is provided together with a brief introduction to the principles of multi objective optimisation. The chapter then introduces the aims and objectives of the research before closing with a description of the remainder of the thesis.

1.1. Background

Every day of their lives people make decisions - some of which are made consciously while others are made unconsciously. Examples of decision-making range from the personal and (more often than not) insignificant task of deciding what clothes to wear through to the more complicated professional task of product design. The first of these two tasks usually requires little effort. The latter however can take years and involve changing priorities over time. Whilst the level of complexity differs in an order of magnitude the fact remains that they can both be considered examples of an optimisation process. The first optimises a person's appearance whilst the second optimises the price, functionality and ergonomics of the product.

The process of optimisation will involve the weighing up of various alternatives, the resolution of conflicts between the criteria and the discarding of invalid options in order to arrive at the optimum solution. A classic example of this can be seen in the knapsack problem (Kellerer *et al*, 2004). The knapsack problem derives from various fields and can be stated in single criteria form as "Given a set of items, each with a cost and a value, determine the number of each item to include in a collection so that the total cost is less than some given cost and the total value is as large as possible". When the problem is considered as one of multi criteria each item might be assigned an

additional factor such as ‘appeal’ and stated as “Given a maximum weight limit and physical collection size, together with a series of objects with criteria such as weight, size and appeal. What is the optimal set, measured as the number of items having the most appeal, of objects that may be carried without breaking the maximum weight limit or physical size of the collection?” Other examples of theoretical objective optimisation problems can be seen in the travelling salesman’s scenario (Applegate *et al*, 2007) which states that “Given x number of cities, which is the shortest route, that visits each city only once then returns to the source”. Variations remove the requirement to return to source.

The travelling salesman problem can be described as a single objective path problem with constraints. The shortest distance is the objective and single city visit (or avoidance of repeat visits to the same city) being the constraint. Humans normally tackle decision problems like the two examples given above by attempting to find the solution representing the best compromise between the criteria. As multiple criteria are being evaluated such problems are considered “Multiple Objective Problems” (MOPS). While across applications the variables to be optimised change the basic task of optimisation does not. It “will involve the application of a great deal of experience, knowledge and an ability to weigh up potentially large numbers of possibilities. This process becomes harder and often intractable as the number of decisions required and the system or product complexity increases.” (Todd, 1997 p.2). As the number of criteria that require optimisation increases so does the time, effort and complexity in doing so using traditional computing algorithms. Although any given solution to such a problem may be verified quickly there is often no known efficient way to locate a high quality solution in the first place (Garey and Johnson, 1979). During the process of optimisation an assessment as to the quality of a given solution has to be made. As an example in the product design problem, a change might reduce the weight of the product but increase the size and cost whilst in order to decrease the cost of the product both the size and weight may need to be varied. Suman (2004, p.1849) states that “a good multi-objective optimisation algorithm must find a set of solutions without biasing any objective”. Luger (2002) suggests that one of the key factors in any multi-objective

analysis is having the ability to distinguish between a good, useable solution and a poor one. Goldberg (1989) highlights that historically several approaches to the solution of multiple objective problems have been identified including enumerative, deterministic and stochastic methods.

Algorithmic methods such as linear programming have long been used for solving mathematically based problems such as the shortest path problem. However, these techniques lack the robustness and capacity required for effectively solving problems with multiple criteria (Coello-Coello and Lamont, 2005). Where such techniques are used in a “brute force” manner the time required to optimise the problems becomes infeasible. Alternative methods involve the simplification of the process such as reducing the problem from multiple criteria problems into single criteria problems. Techniques from artificial intelligence however allow for the production of a set of “compromise” solutions for a given optimisation problem. The work undertaken aims to investigate some of these techniques when applied to the path planning process.

1.2. The Path Planning Problem

The task of path planning is the process of finding the most effective route from a given start point to an end point. Traditionally this has been based upon the least cost or shortest path and generated using linear algorithms such as the Dijkstra (1959) shortest path algorithm where a single criterion, typically distance or travel time is used to determine the “shortest” path through a network. Martins and de Santos (1999) highlight that whilst single criteria algorithms such as the Dijkstra (1959) algorithm have been the focus of a great deal of research comparatively little attention has been focused on the optimisation of the path planning process when more than a single criterion is involved. Where such work has been undertaken the typical view has been that the optimal path will simply be the shortest path with the lowest total cost across the sum of the individual criterion or importance suggested through the use of weightings to give preference to a criterion. Figure 1.1 and Figure 1.2 present a graph with single criterion associated with each edge and the same graph with three criteria

values associated with each edge. Table 1.1 and Table 1.2 demonstrate the single criterion and three criteria shortest path from the vertices 1 to 4 respectively. Table 1.3 demonstrates the effect of the application of various weightings on each criterion in order to suggest the application of relative importance to each criterion on the ranking of results. In Equation 1.1 the simple multi criteria weight system is highlighted. It should be noted of course that in many cases the application of weightings to a multi objective function might be more than adequate. However, the reduction in effort involved in the implementation may often be outweighed by the effort involved in generating ideal weights which have the potential to vary from person to person.

$$f = \sum_{i=1}^D C^i W^i$$

Equation 1.1 Simple Weighting

Where D is the number of criteria, C is the fitness cost associated with a given solution objective and W is the weighting assigned to that objective.

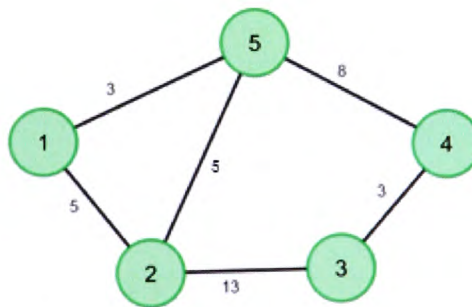


Figure 1.1 Simple Example of a Single Criteria Graph

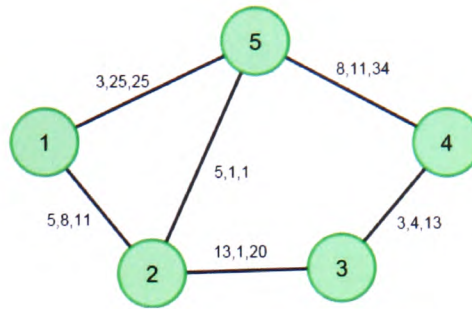


Figure 1.2 Simple Example of a Multi Criteria Graph

Path Vertices	Total Cost
1,5,4	11
1,2,5,4	18
1,2,3,4	21

Table 1.1 Shortest Path Across Single Criteria Graph

Path Vertices	Total Cost
1,2,3,4	78
1,2,5,4	84
1,5,4	106

Table 1.2 Shortest Path Across Total Sum Graph

Path Vertices	Weights	Criteria Cost	Total Cost
1,2,5,4	{10,0.3,11}	{180,6,506}	692
1,2,3,4	{10,0.3,11}	{210,3.9,484}	697.9
1,5,4	{10,0.3,11}	{110,10.8,649}	769.8

Table 1.3 Shortest Path Across Weighted Sum Graph

1.3. Applications of Path Planning

The field of path planning has attracted researchers from a wide range of disciplines due to the wide range of applications benefiting from the process of network optimisation. Ahuja *et al* (1993) argue that networks can be seen in virtually every aspect of day-to-day life in the form of transportation networks, telecommunications networks and even social networks. A brief scan of the literature on the process of network optimisation has identified several application areas, notably:

1.3.1. Transportation Networks

Many automotive manufactures now offer integrated GPS and GIS systems for vehicles to provide on board routing information. Cities have linked GPS systems with the public transport systems via real time displays at bus stops and the World Wide Web (Cardiff 2007). Coutinho-Rodrigues *et al* (2012) present a multi criteria GIS based application for the evacuation of city areas during emergencies. Saadatseresht *et al* (2009) present a similar system for evacuation of building groups such as a campus. Apple (2012) has proposed the use of crowd-sourced information to detect traffic congestion and allow rerouting along less congested routes. Solutions to the urban transit network design problem (UTNDP) are presented in Fan and Mumford (2010) where the authors present solutions to the urban transit problem by applying a weighting function to the total distance and number of transfers required to complete a journey using public transport.

1.3.2. Computer Networks

The ability to provide quick and reliable network routing information across modern computer networking facilities can be seen as central to the optimal performance of wide area networks such as the internet. This is especially true given the dynamic nature of these networks where the presence and cost of traversing certain links can change rapidly (Wen *et al* 2007; Kauer *et al* 2003). The work of Chitra and Subbaraj (2012) makes use of a Genetic Algorithm for multi objective path planning in computer networks. Gen *et al* (1997, p.401) produce a Genetic Algorithm approach to the dynamic routing problem on networks stating “The purpose is not, of course, to compare the Genetic Algorithms with conventional algorithms, because Genetic Algorithms will be unable to compete”. In that work the authors are handling single criteria shortest paths using a Genetic Algorithm. The authors report reasonable levels of success of graphs of limited size (vertices 70, edges 211).

1.3.3. Fighting Organized Crime

Furtado *et al* (2009) combine heuristic techniques with graph theory to develop a crime analysis model. Xu and Chen (2004, p.473) also demonstrate how network analysis can be used to identify associations in criminal networks. That work attempts to model the connections between offenders such as “kinship, friendship, co-workers or business associates”. Flores *et al* (2012) highlight the use of social networks during the ‘Arabic spring’ of 2011. The authors question whether graph centrality measures could be used to prevent unwanted ‘collective action’ such as terrorism. Zengen and Mao (2007) review the issue of money laundering.

1.3.4. Medical Applications

Aittokallio and Schwikowski (2006) apply graph-based algorithms to the identification of networks and clusters in cell biology. Chen *et al* (2009) present semantic graph operations based upon directed graphs to identify disease-causal genes. They highlight links between elements on a semantic graph.

1.4. Decision Making in Path Planning

Many real world networks such as roads can be based on geographical data with the cost metric being considered as an accurate measure of actual distance between vertices. Additional costs can be associated with any number of metrics including, but not limited to, the maximum speed limit of the road, monetary cost of traversal, estimated CO₂ output and the number of traffic lights or junctions encountered. The ability of software based algorithms to quickly process the connectivity information contained within graph structures and calculate the least cost path is something which when completed manually could take many orders of magnitude longer to compute to the same level of accuracy. Ahuja *et al* (1993), together with a great number of introductory texts on software engineering and computer based data structures provides an overview of the basic methodology and algorithms used to calculate a least cost path.

The historic, that is to say least cost, approach to the route optimisation process fits well with the generally argued view that at an instinctive level people will attempt to follow the shortest path forming a route. However this contrasts with the view of Duckham and Kulick (2003 p.3) who state “several cognitive studies have shown that people prefer the simplest path”. Rickter and Duckham (2008) present an algorithm based upon the simplification of instructions rather than the route itself. Burgess and Darken (2004 p.1) state that there is a clear preference for “lines of drift” indicating that whilst people prefer a shorter route they are fully prepared to travel further for a perceived simpler route. Liu *et al* (1994) suggests that in many cases the shortest path

algorithm is not always the best method of route planning with the authors suggesting that using only these algorithms may also produce solutions that are not suitable for human drivers, “For example, human drivers would normally like to drive on major roads” (Liu, 1996 p.2). Car (1997) presents a hierarchical routing algorithm to assist in routing optimisation systems in order to achieve such solutions where preference is derived to higher speed and capacity road links such as motorways or dual carriageways. Bailenson *et al* (1998) highlight that users often choose routes consisting of the longest and straightest road links while Li *et al* (2010) suggests that drivers prefer a ‘simple driving’ route which may be longer in time, distance or any other factor but is perceived to be easier to travel along at certain times. Later in Liu *et al* (1994) it is argued that in many cases the optimal route is often dependant on the individual. A user’s empirical knowledge of the route they are going to be travelling defines the routes optimality. Li *et al* (2005) find that 60% of tested users regularly choose between travelling along multiple routes connecting the same locations depending on levels of congestion. Lyons *et al* (2008) present a 2 dimensional caricature of decision makers identifying two groups, those who would like as much information as possible (the Mr. Spock approach) through to those who believe that provided the destination is reached the route itself is not an issue (the Homer Simpson approach). People have highly embedded travel habits and the nature of these habits may limit or enhance the need for travel information. SRA (2005) suggest that too much emphasis is given to the notion that travel choices are made as a ‘staged approach’ and instead highlight that a person may base a decision on personal needs or circumstances. Davies and Lingras (2003, p. 31) state “A person not only relies on a single favourite route, but also several alternatives. At any given time, an appropriate route may be chosen by splicing together sections of all routes in mind, depending on the network conditions at that time. Without the aid of an algorithm and ability to process the entire network information a person may not use the optimal path”. Bonsall (1992) undertakes an analysis into the effectiveness of route guidance systems on individuals’ behaviour. Decisions regarding journey choice in that work highlight that only 35% of all journeys complied with the advice given and in part depend on a range of other factors such as age or number of miles driven per annum and the perceived quality of advice received in the past.

The work of Bonsall (1992) links to a phenomenon in the field of discrete choice theory: satisficing behaviour (Miller and Star, 1967). This is not a new theoretical phenomenon but it is now being acknowledged in more recent travel information literature (Chorus *et al*, 2006; Lyons, 2006). Satisficing behaviour concerns an individual being prepared to select a travel option that meets their minimum requirements (is ‘good enough’), even if other options exist which may be better (but which could require additional effort to identify). Papinski *et al* (2009) introduce the work of Golledge and Stimpson (1997) in which the concept of a ‘knowledge base’ is considered. Route choice decisions are largely based on existing knowledge and experience that shapes the evaluation of choice alternatives and develops into concept of what is perceived to be an optimal route or set of optimal routes. Papinski *et al* (2009) highlight how route selection is often a two stage or issue process. The first issue involves route learning through the identification of key landmarks prior to the trip. A second set of route choice decisions are made while en-route which involves information processing. Duckham *et al* (2010) introduce an algorithm which relays local landmark information as part of the selected journey routing information with selected landmarks made part of the instruction set. Papinski and Scott (2011) introduce a geographical information system based analysis into how routes users actually travelled differ from those suggested based on criteria a single criterion such as shortest distance or time. The authors present several cases where the route travelled varies significantly to those produced based on path analysis with users spending more time on highways than was suggested based on either the optimisation of time or distance confirming the previously discussed proposition by Liu *et al* (1996). Quattrone and Vitetta (2011) perform a similar study where the authors make use of GPS acquired route information to determine the validity of a fuzzy route choice model. Azaria *et al* (2012) present a reward based mechanism which attempts to ‘trick’ a driver into following the optimal rather than the preferred routes.

Acuna and Parada (2010 p.1) make a direct comparison between computationally derived solutions to an NP-Hard path planning problem (in the form of the travelling salesman problem) to those solutions produced by human participants. The authors state “Humans need to solve computationally intractable problems such as

visual search, categorization, and simultaneous learning and acting, yet an increasing body of evidence suggests that their solutions to instantiations of these problems are near optimal". Bekhor *et al* (2006, p. 235) suggest that in the 'real world' the selection of sub optimal paths is logical given that drivers "have imperfect knowledge of traffic conditions and limited information processing abilities".

Recent years have seen a huge increase in the availability of navigation information. In 2006 Google™ introduced online a mapping application with the ability to provide point-to-point directions to users. Mayer (2011) suggests that 12 billion miles of routing information are provided by the application every year. Of importance to this work given the previously discussed approaches to path planning is the applications' ability to allow users to manually redirect a route to match their particular preferences and manually update the associated routing information in line with those user selected route changes. Simultaneous to increased availability of online mapping has been the use of global positioning system (GPS) based hardware. Berg-Insight (2012) suggest that as of 2011 there are over 340 million GPS enabled devices in use world-wide with 33 million personal navigation devices being sold in 2011 alone. A US based study by Harris Interactive (2007) suggested that 81% of respondents found the ability to automatically recalculate routes taking into account driver error useful with real time traffic updates being 'useful' to 75% of users. It should be noted that the routes provided by both on-line mapping applications and personal navigation devices (PND) will often be contrary to the models of path choice seen in the literature. The provided routes represent optimal routes in terms of travel time, distance or road choice but as shown by Papinski and Scott (2011) are not considered optimal by users.

The discussion regarding route choice has to this point considered only personal travel. However for freight transport the emphasis on fastest (either in terms of distance or time) routes is not always in the interest of the driver and may have negative consequences when the routes run through built-up areas. Existing road data sets often lack the completeness required for freight management where a larger vehicle may not be able to

navigate certain turns or bridges etc and so ‘off the shelf’ navigation devices are unsuitable for use in the freight industry. A study by Arentze (2012) suggests that drivers of larger sized trucks avoid urban areas due to the difficulty in navigating busy streets while the drivers of smaller vehicles often, as is the case with car drivers, follow what they consider to be a shorter path regardless of the fact that it may not in reality be the case. Hubschneider (2012 p. 494) presents a routing model specifically aimed at freight transport that produces routes “which are often are longer but comparable in time and fuel consumption”.

1.5. Research Definition

There are many well understood algorithms that can be used to generate the shortest path between any two points on a network including the Dijkstra (1959) algorithm and the A* algorithm. However despite the amount of research that has been carried out into the single criterion problem comparatively little effort has been placed on the problem of route optimisation that involves more than a single criterion. Where research into the topic has been undertaken it has frequently involved the condensing of multiple criteria into a single criterion or involved a limited number of criteria.

The research undertaken focuses on the development and comparison of a number of AI based techniques including Genetic Algorithms, the Tabu Search technique and Simulated Annealing in an attempt to provide a wider range of techniques to be used in the process of multi-criteria graph optimisation. This area of work covers a myriad of issues but attempts to address two core issues, namely the analysis of the multi-criteria algorithms and methodologies when considering the optimal path problem together with the development and analysis of techniques to achieve these optimal solutions(s). As has been indicated throughout this chapter, it is agreed that the route and path planning process is one that is inherently multi objective in nature. However little work has been undertaken in the use of various optimisation

processes with the aim of solving such problems in a truly multi objective way. Mooney (2004) provides one of the few exceptions to this, developing a substantial study into evolutionary algorithms applied to the process of multi objective shortest path analysis.

1.6. Research Aims

The shortest path problem where multiple criteria are considered is an example of a problem that is NP-Hard (Granat and Garrerio, 2003). The identification of optimal solutions on anything but very small graph using brute force techniques is not viable given the intractability of the problem. Mooney (2004) addresses the issue using heuristic functions in the form of evolutionary algorithms as do Saadatseresht *et al* (2009), Liu *et al* (2012) and Cheikh *et al* (2010). However there has been a lack of research interest in applying other heuristic approaches such as Hill Climbing (Russell and Novig, 2003), the Tabu Search (Glover and Laguna, 1987) or Simulated Annealing (Kirkpatrick *et al*, 1983) to the Multicriteria shortest path problem (MSPP). This is despite the fact that other heuristic approaches have been applied to graph related problems in the past such as the metro map layout problem (Stott *et al*, 2011) and the variations (in the form of multiple objectives) to the travelling salesman problem. It is the principle aim of this work to consider alternative heuristic techniques in the solution of the MSPP.

The aims and objectives of the research are:

- To develop alternative (to the Genetic Algorithm) heuristic techniques for the solution of the MSPP

- Assess the ability of those heuristic techniques to solve the MSPP against real world and synthetic graphs

- Compare the alternative heuristic approach with algorithmic methods for the solution of the MSPP

1.7. Thesis Format

The remainder of this thesis is separated into six chapters. Chapter Two reviews the terminology used in graph and network theory. It then proceeds to review the data structures and algorithms that can be used in the processing of graph connectivity information. One of the principle algorithms for the calculation of single criteria shortest paths in the form of the Dijkstra shortest path algorithm is presented together with a brief description of the various data structures used to enhance the performance of that algorithm. Other shortest path algorithms are briefly introduced. The chapter then attempts to formalize the issue of the MSPP before discussing the various existing methods that can be seen in the literature.

Chapter 3 reviews the various heuristics under consideration as part of this thesis. Genetic Algorithms, the Tabu Search and Simulated Annealing are introduced and existing methodologies for solving multi objective problems are considered. The chapter concludes with a discussion of the various quality metrics applied to multi objective problems. Chapter 4 introduces the algorithms used in the experimental phase of the research problem. The test data sets employed in study the assessment of those algorithms are discussed as are the method used in the acquisition and preparation of those datasets for the experimental phase of the project. In Chapter 5 the runtimes seen in the algorithms are considered together a summary of the quality results. The chapter ends with a consideration of the optimal choice of algorithm under various circumstances. Chapter 6 presents a more detailed analysis of the mode of operation for

each of the algorithms. Possible limitations of the experimental phase of the work are also considered. Finally, Chapter 7 concludes the project and discusses potential avenues of future work.

Chapter Two: Graph Theory and Shortest Path Analysis

2. Graph Theory and Shortest Path Analysis

The chapter opens with an introduction to graph properties before moving to provide an overview of the basic terminology relating to graphs. The chapter then proceeds to introduce single criteria path optimisation and in doing so discusses variations to the shortest path problem such the K shortest path problem. More recent methods for the solution of shortest path analysis, in effect those forming the state of the art in shortest path analysis, are then considered before considering the analysis of social networks. The chapter then turns its attention to the concepts of Pareto optimality introduced through a worked example of the MSPP before discussing existing methods for the solution of the MSPP.

2.1. Background

Graph based structures can be found in many real world applications ranging from transportation through to chemistry and increasingly on-line gaming and social networking (Newman *et al*, 2002). Other examples of graph-based structures exist in the form of computer networks where the network presented is often dynamic in nature. Ahuja *et al.* (1993, p.2) argue that graphs and networks can be seen “Everywhere we look”. Table 2.1 overleaf provides examples from real world networks. Pallotino and Scutella (1997) claim that since the end of the 1950s over 2,000 pieces of literature on the process of graph optimisation have been published.

Figure 2.1 (Sedgewicke, 2003) presents an overview of various graph types highlighting the variations in the connectivity and geometric properties of the different graph structures. In Figure 2.1 a series of graph types ranging from complete, random, grid, real world and small world graphs are presented. Due to the general pervasiveness of graph optimisation in the literature this work will not delve into the matter in any great detail. A comprehensive study into the field can be found in Ahuja *et al* (1993).

Other notable works can be seen in Merris (2000), Ford and Fulkerson (1962) and Begre (1973). The following section briefly introduces the graph terminology used throughout the remainder of this work. The chapter then proceeds to provide an overview of both the single and multi criteria path optimisation process.

<i>Applications</i>	<i>Physical Vertices</i>	<i>Physical edges</i>	<i>Flow</i>
Communication Systems	Telephone exchanges Computers Satellites	Cables Fiber optic links Relay links	Voice Messages Data
Hydraulic Systems	Pumping Stations Lakes	Pipelines	Water Oil
Integrated Computer Circuits	Gates Registers	Wires	Electricity
Mechanical Systems	Joints	Rods	Heat Energy
Transportation	Airports Rail Yards	Highways Airlines Routes	Passengers Vehicles

Table 2.1 Applications of Graphs (Ahuja et al, 1993)

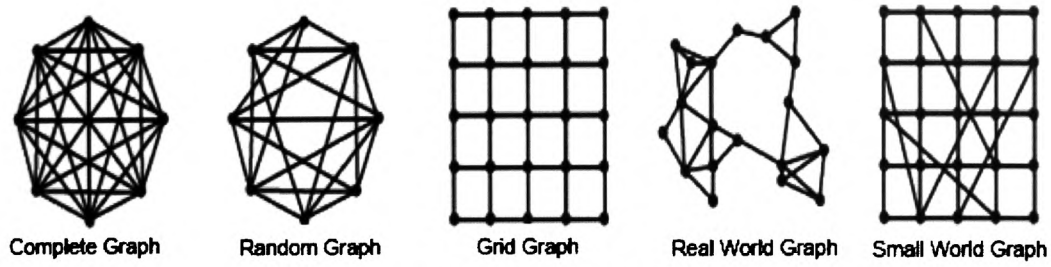


Figure 2.1 Various Graph Types (Sedgewicke, 2003)

2.2. Terminology

The following section introduces the terminology used throughout the remainder of this work. The terminology will then remain constant throughout.

Definition 1: Graphs

A graph $G = (V, E)$ consists of two sets, V and E . The elements of set V are called vertices (or nodes). The elements of set E are called edges (or arcs). Each edge has a set of one or two vertices associated with it that are called its endpoints. An edge is said to join its end points. A self-loop is an edge which joins a single endpoint to itself in a loop.

If vertex v is an endpoint of the edge e , then v is *incident* to e and e is *incident* on v . A vertex v is adjacent to vertex u if they are joined by an edge. Two adjacent vertices are considered neighbours. Adjacent edges are those edges with share a common endpoint. A simple edge occurs when only a single edge occurs between two endpoints. A multi-edge is a collection of two or more edges sharing the same endpoints. The degree, $d(v)$, of a vertex v is the number of edges in set E that have v as an endpoint. A degree sequence is a non- increasing sequence of vertex degrees. The density of a graph

is the ratio of the number of vertices and edges. Figure 2.2 presents an example of a simple graph in visual form.

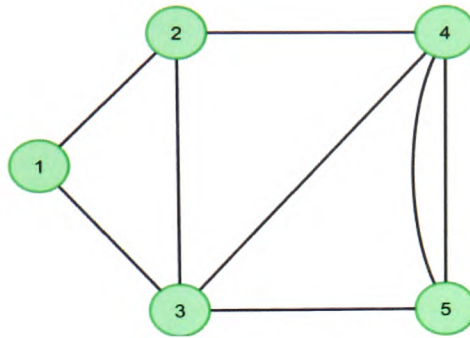


Figure 2.2 Simple Graph Without Costs

A directed graph (or digraph) consists of a number of directed edges. A directed edge is an edge $e(i,j) \in E$ one of those endpoints is designated the tail while the other is designated as the head of the edge. The edge leads from the tail to the head. Figure 2.3 presents a directed view of the graph presented in Figure 2.2. A weighted graph has attribute information associated with each edge in the graph called the edge costs and is shown in Figure 2.4

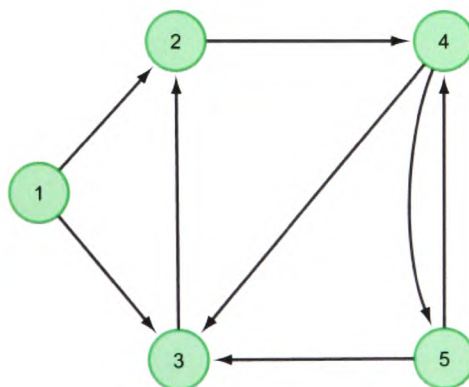


Figure 2.3 Simple Directed Graph

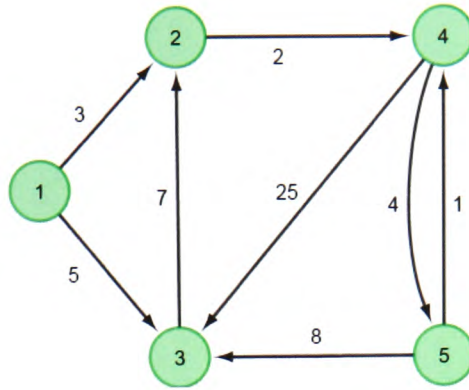


Figure 2.4 Simple Directed Graph with Costs

Definition 2: Walks and Paths

A walk in the graph $G = (|V|, |E|)$ is a sub graph of G consisting of a sequence of vertices leading from one vertex (u) to another (v) such that for $w=1, \dots, n$, the vertices V_{w-1} and V_w are adjacent endpoints of edge e .

$$W = \{w_0 = u, w_1, \dots, w_n = v\}$$

A valid path in the graph $G = (|V|, |E|)$ is a walk without the repetition of any vertex in that walk. The length of a path is given by the number of edges making up that path. A geodesic path across is the shortest path (in terms of edges traversed) between two vertices. The diameter of a graph is the largest geodesic path seen. The radius of a graph is the shortest geodesic. A Path Description Vector (PDV) details the costs of traversing a path. Element k of a path description vector for path $W_{s,t}$ indicates the total sum value of the criteria $k \in D$ over the path $W_{s,t}$ from s to t where D is the number of costs or weighting associated with an edge. Figure 2.5 shows a PDV between the vertices one and five using the graph presented in Figure 2.4 as its basis where the value

of $D = 1$. A formal presentation of the PDV where multiple criteria are considered is given in Equation 2.1 where $D=4$.

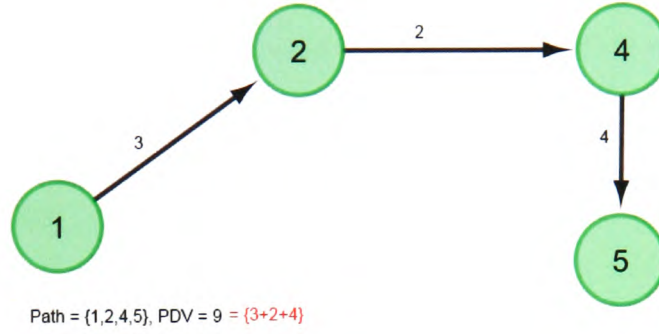


Figure 2.5 Example Path Description Vector (PDV)

$$PDV = \begin{cases} C1 = \sum_{e(i,j) \in W(s,t)} c_{i,j}^1 \\ C2 = \sum_{e(i,j) \in W(s,t)} c_{i,j}^2 \\ Ck = \sum_{e(i,j) \in W(s,t)} c_{i,j}^k \\ CD = \sum_{e(i,j) \in W(s,t)} c_{i,j}^D \end{cases}$$

Equation 2.1 The Path Description Vector

Where $e(i,j)$ is an edge present in the path $W(s,t)$

And $c_{i,j}^k$ is the weight associated with an edge in criteria k.

Definition 3: Random Walks

A random walk consists of a walk through the graph G . Random walks are generated using a random selection mechanism where the number of outgoing edges from a vertex is greater than one. The process of iteration at each vertex in the walk continues until the target vertex is reachable. Aldious and Fill (1999) undertake an in depth study into the principles of random walking. Of particular interest to this thesis is the application of repeating random walks to enumerate paths across the graphs. Zijpp and Catalano (2005) suggest a possible alternative to the random method of enumerating paths. In their study a constraint-based approach to the k-shortest paths algorithm is used where limitations of the upper and lower PDV values act as constraints. In recent years random walks have been proposed in the context of querying and searching (Avin and Bretto, 2004), routing and self-stabilization in wireless ad-hoc networks (Dolev *et al*, 2002, Servetto and Barrenechea, 2002) and peer-to-peer networks (Gkantsidis *et al* 2004). Elsässer *et al* (2011) study the time taken to cover a graph using multiple instances of a random walk as do Alon *et al* (2008). Both works attempt to run several walk instances in parallel. Alon *et al* (2008) report a linear increase in coverage time to the number of walk instances initiated.

Definition 4: Multi Objective Shortest Path Problem

The aim of the multi objective shortest path (MSPP) problem is to identify those paths between two vertices in a graph $(G = (V, E))$ in the set $\rho(s, t)$ of valid paths between two vertices where the PDV is minimised. Equation 2.2 shows the formal function of the MSPP.

$$\text{Minimize } (PDV\{C^1, C^2, C^k, \dots, C^D\} \forall W_{(s,t)} \in \rho(s, t))$$

Equation 2.2 The Multi Objective Shortest Path Problem (MSPP)

With the following constraints:

$W(s,t)$ is a path with no repeating vertices

$W(s,t)$ is valid with each edge $e_{i,j} \forall E(G)$

D = The number of criteria

Definition 5: Centroid

The term Centroid is used to describe the centre point of a feature. Figure 2.6 presents a visualization of a Centroid as used in later sections of this work (see Chapter 4) where a real world coordinate is used to describe the centre point of a regular geographic area.

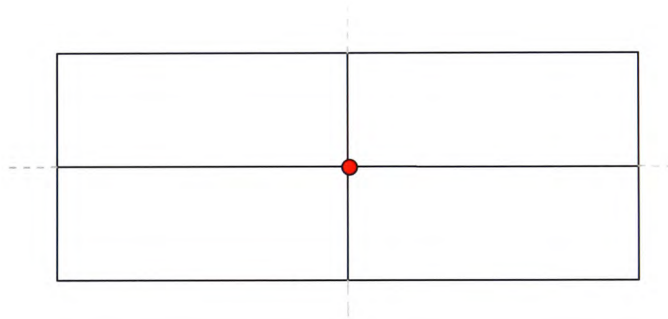


Figure 2.6 Centroid of a Rectangle

2.3. Random Graph Generation

Random networks are frequently generated to investigate the effects of model parameters on network properties to test the performance of network analysis algorithms. Chapters 5 and 6 of this work make use of such graphs to analyse the performance of various meta-heuristic algorithms when solving the MSPP. An in-depth study of random graph generation techniques is beyond the scope of this work however

this section introduces a series of prominent approaches to random graph generation seen in the literature. The following methods graph generation are introduced:

- The Erdos and Renyi Model
- The Gilbert Model
- The Barabási–Albert scale free model

2.3.1. The Erdos and Renyi Model

The theory of random graphs was founded simultaneously in Erdos and Renyi (1959) and in Gilbert (1959). What sets both works apart is the probabilistic approach to random graph generation employed.

Erdos and Renyi set out to investigate what a typical graph with V labelled vertices and E edges ‘looks’ like. In Fowler *et al* (2009) the model is used to generate sample datasets to test the connectivity between individuals in social networks where the model is used to produce highly connected networks. Ioannides (2006) also considers the networks produced by the model as being the upper bound set of connectivity in models of social networking. Both the works of Fowler *et al* (2009) and Ioannides (2006) highlight that the models produced are not realistic models of those seen in the real world. Algorithm 2.1 introduces the model in high level (and simplistic) pseudo-code form.

The model considered by Erdos and Renyi is an appropriate method for generating random graphs with a fixed number of vertices and edges where the likelihood of an edge between any two vertices in the set V being inserted into the graph being of equal probability to any other pair of vertices. Variations to the model introduced in Austin *et al* (1959) allow for the introduction of parallel edges in the graph. Figure 2.7 presents an example graph $G = (5, 8)$. The graph structure included in Appendix A.

Algorithm:	Erdos-Reyni Random Graph Generation Model
Input:	V = Number Of Nodes In Graph E = Number Of Edges In Graph
Output:	A Graph $G=(V , E)$
$EDGES$ = List Of Existing Edges = {} $e(i,j)$ = An Edge Between Two Vertices	
<pre> FOR (i = 0; i < E ; i++) { s = Select Edge Source At Random From {1,2,.. V } t = Select Edge Target At Random From {1,2,.. V } WHILE ((e(s,t) MEMBER OF EDGES) { s = Select Edge Source At Random From {1,2,.. V } t = Select Edge Target At Random From {1,2,.. V } } EDGES = EDGES + e(s,t) } </pre>	

Algorithm 2.1 Erdos-Reyni Random Graph Generation Model

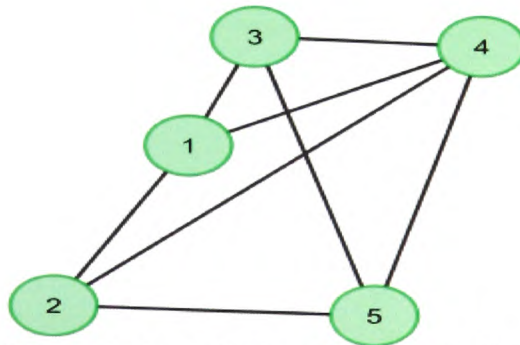


Figure 2.7 Example Graph Generated using Erdos-Reyni Model

2.3.2. The Gilbert Model

Unlike the model of Erdos and Renyi the Gilbert model of random graph generation does not guarantee any pre-designated number of edges will be present in the graph. In Gilberts' model, the $\frac{|V|(|V|-1)}{2}$ potential edges of a simple undirected graph with $|V|$ vertices are included with the probability $0 < p < 1$. As demonstrated in Algorithm 2.2 the Gilbert model is from a computational perspective a simple process. Despite the computational simplicity of the model Batagelj and Brandes (2005) highlight that the methodology is unsuitable for the generation of large, sparse graphs. The model operates with a run time in the order of $\Theta(n^2)$. Many works in the literature refer to the Gilbert model as a variation on that of Erdos and Reyni with the later introducing the model in their work of 1959 simultaneously with Gilbert. The model is referred to here as the Gilbert model for clarity purposes only.

Algorithm:	Gilbert Random Graph Generation Model
Input:	V = Number Of Nodes In Graph P = Edge Probability $0 < p < 1$
Output:	A Graph $G=(V , E)$
$EDGES$ = List Of Existing Edges = { }	
$e(i,j)$ = An Edge Between Two Vertices	
<pre> FOR (i = 1 TO V) { s = i FOR (j = i TO V) { t = j Generate a uniform random number $\theta \in \{0, 1\}$; IF ($\theta < P$) $EDGES = EDGES + e(s,t)$ } } </pre>	

Algorithm 2.2 Gilbert Random Graph Generation Model

Figure 2.8, Figure 2.9 and Figure 2.10 demonstrate the output of the Gilbert model of graph generation with three probability values, those being 0.1, 0.6 and 0.8. The number of nodes in each graph ($|V|$) is set to five prior to the generation of the graph. The aim is not to produce instances of large complex graph but rather to show the effect of the probability value on the graph output. As the probability value p increases from 0.0 to 1.0 the likelihood of encountering a disconnected vertex decreases. The graphs structures are included in Appendix A. As seen in Figure 2.8 at low probability values there are a limited number of edges presents in the graph.

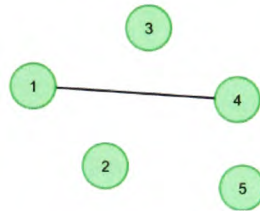


Figure 2.8 Example Graph Generated using the Gilbert Model ($p=0.1$)

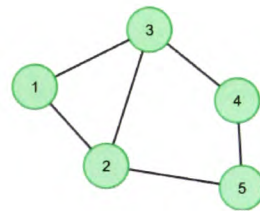


Figure 2.9 Example Graph Generated using the Gilbert Model ($p=0.6$)

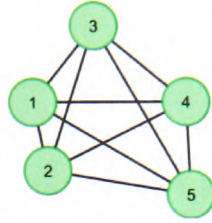


Figure 2.10 Example Graph Generated using the Gilbert Model ($p=0.8$)

2.3.3. The Barabási–Albert Scale Free Model

Milgram (1967) introduced the concept of small worlds. The work highlighted a limited sized path could be found to connect any two individuals in differing US cities. Watts and Strogatz (1998, p.440) suggest that similar properties can be seen in a variety of other real world graphs or networks stating, “Ordinarily, the connection topology is assumed to be either completely regular or completely random. But many biological, technological and social networks lie somewhere between these two extremes”. Watts and Strogatz highlight two structural properties of what are commonly referred to as ‘small world’ graphs, firstly the typical path length connecting any two graph vertices will be low and secondly that the graphs will demonstrate a high level of ‘cliquishness’ where a subset of the vertices in a graph will demonstrate a higher degree than others. Newman (2001) highlights a limited community structure between US based scientists as being an example of a small world network.

Watts and Strogatz introduce a model for generating small world graphs. The model is initialized with what may be able to be visualised as a ring lattice consisting of $|V|$ nodes. Each node connects to K neighbours ($K/2$ on either side). The model then proceeds to rewire, at random, each edge of the lattice with probability p such that self-connections and duplicate edges are excluded. This process introduces long-range edges which connect nodes that would otherwise be part of alternative ‘neighbourhoods’. The variation of p through $0.0 - 1.0$ enables the modification of the graph between ordered

(0.0) and randomised (1.0). Albert and Barabasi (2002) highlight that the model of Watts and Strogatz introduces an unrealistic vertex degree distribution. Figure 2.11, Figure 2.12 and Figure 2.13 present a simple graph with $V = 10$ nodes with varying values of $p = 0.0, 0.5$ and 0.9 respectively. $K=4$ in all three examples. The graph structures have been included in Appendix A.

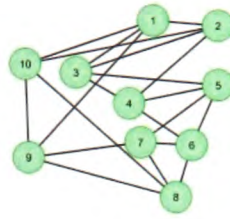


Figure 2.11 Example Graph Generated using the Watts and Strogatz Model ($p=0.1$)

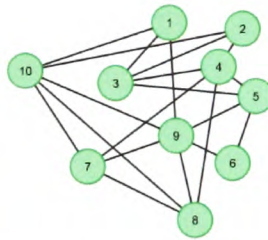


Figure 2.12 Example Graph Generated using the Watts and Strogatz Model ($p=0.5$)

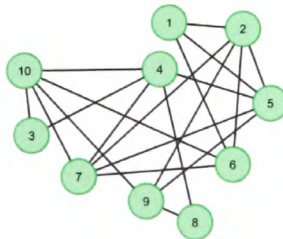


Figure 2.13 Example Graph Generated using the Watts and Strogatz Model ($p=0.9$)

The model of Barabasi and Albert (1999) seen in Algorithm 2.3 in pseudo-code form aims to overcome the issue of an unrealistic degree distribution seen in the Watts and Strogatz model. The authors argue in Albert and Barabasi (2002 p. 71) that models such as the Watts and Strogatz model are limited by the definition of a predefined number of vertices. The model generates the small-world property through the rewiring of the graph at random and without considering the creation of new vertices in the graph or the degree of existing edges in the graph. The ‘scale-free’ graph generation model of Barabasi and Albert starts with a small number ($m\theta$) of nodes in the graph. At each iteration (x), the model adds a new node with $m(\leq m\theta)$ edges that link the new node to m edges already present in the network. The probability of an edge generation in the network is equal to the probability outlined in Equation 2.3 where d_i is the existing degree of vertex i .

$$p(d_i) = \frac{d_i + 1}{\sum_j d_j + 1}$$

Equation 2.3 Barabasi and Albert Probability Value

After x iterations the model will have developed a network consisting of $V = x + m\theta$ vertices and mx edges. Albert and Barabasi (2002) highlight that over time the degree distribution will start to decay or the graph will become completely connected. Figure 2.14 (the initial graph) and Figure 2.15 (following the graph growth procedure provide an example of a simple graph generated using the Barabasi and Albert model with $m\theta = 4$ and $V = 10$.

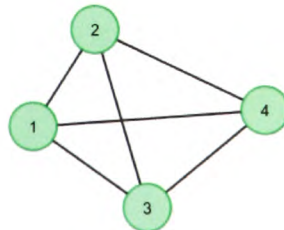


Figure 2.14 Example Graph Generated using the Barabasi and Albert Model ($m\theta=4$)

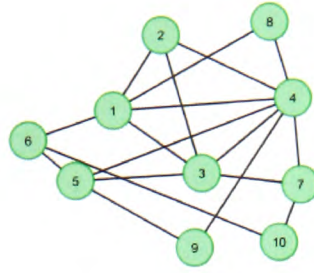


Figure 2.15 Example Graph Generated using the Barabasi and Albert Model (v=10)

2.3.4. Alternative Methods of Random Graph Generation

The methods of graph generation considered to this point can be considered seminal works in the field of random graph generation. The introduction to the current review of random graph generation highlighted that a detailed study of the research area is beyond the scope of this work to consider completely however in the current section a number of other methods of graph generation are briefly considered.

Tobita and Kasahara (2002) introduce the ‘Layer-by-Layer’ method of random graph generation. In many ways is an extension of the Gilbert model where edges are added to the graph with a pre-designated probability. The method splits the graph into a number of separate layers with the number of vertices in each layer being equal to $LV = \frac{V}{L}$. Nodes are randomly assigned to a specific layer with edges between layers generated using the probability method developed by Gilbert. The graphs produced by the structure have been used in critical path analysis. Bayati *et al* (2007) introduce a random generation model for graphs with a prescribed degree sequence. The model starts with an empty graph structure and is initialized by being a passed the prescribed degree sequence. For the purposes of clarity Figure 2.16 presents a example network with the degree sequence $\{5,3,2,2,2,1,1,1,1,1\}$ and vertices labelled with the degree of that vertex.

Algorithm:	Barabasi and Albert Scale Free Graph Generation Model
Input:	IV = Initial Number Of Nodes In The Graph V = Number Of Vertex To Create In The Graph E = Edges To Add To The Graph
Output:	A Graph $G=(V , E)$
$EDGES$ = List Of Existing Edges = {} $e(i,j)$ = An Edge Between Two Vertices D = List Of Edge Degrees = {} X = Edges Added To Network = 0 // Generate The Initial Seed Network. FOR ($i = 1$ TO IV) { FOR ($j = i$ TO IV) { $EDGES = EDGES + e(i,j)$ $D[i] = D[i] + 1$ $D[j] = D[j] + 1$ $X = X + 1$ } } FOR ($i = IV$ TO V) { A = Number Of Edges Added = 0 $IGNORE$ = Degree Of Edges Ignored = 0.0 FOR ($m = 0$ TO E) { R = Random Number = {0.0-1.0} p = Probability Of Accepting An Edge = 0.0 FOR ($j = 1$ TO i) { IF (Edge Does Not Exist ($EDGES(E(i,j))$)) { $p = (p + D[j]) / ((2 * X) - IGNORE)$ } IF ($R \leq p$) { $EDGES = EDGES + e(i,j)$ $IGNORE = IGNORE + D[j]$ $ADDED = ADDED + 1$ $D[i] = D[i] + 1$ $D[j] = D[j] + 1$ Break From Loop } } } $X = X + 1$ } }	

Algorithm 2.3 Barabasi and Albert Random Graph Generation Model

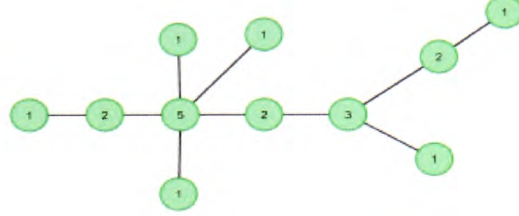


Figure 2.16 Example Network with Prescribed Degree Sequence

Having been passed the prescribed degree sequence the methodology creates $|V|$ disconnected vertices in the graph. The model then sequentially adds edges between two vertices with the probability that an edge is added between nodes i and j proportional to $\widehat{d}_i \widehat{d}_j (1 - \frac{d_i d_j}{4m})$ where \widehat{d}_i and \widehat{d}_j represent the remaining degree of vertices i and j respectively. The remaining degree of a vertex i is equal to d_i minus its current degree. m , the number of edges in the graph is given by $m = \frac{1}{2} \sum_i d_i$, half the total sum of the degree sequence.

Bach *et al* (2012) develop an evolutionary approach to random graphs generation. The approach considers the parameters used in the graph creation process as values to be optimised by an evolutionary algorithm. Rath and Toth (2009) make use the Erdos-Reyni model to generate a graph. They then simulate forest fires caused by lightning strikes to remove clusters of connectivity in the graph. The simulation of fire spreads through the graph with a calculated probability removing random edges based on the probability value. Pennock *et al* (2002) introduces “Winners don’t take all” method. The method extends that of Barabasi and Albert (1999) making use of both

preferential attachments as in the Barabasi and Albert method with a uniform attachment method as an extension. McGlohon *et al* (2008) propose the ‘Butterfly’ model. The butterfly model joins a new vertex to a graph (a), with a randomly assigned variable (P_{step}). An existing vertex (b) on the graph is chosen with the global probability P_{host} . A new edge on the graph (a,b) between the newly generated vertex and the existing vertex is created with an additional global probability P_{link} . The model then traverses P_{step} edges to vertex c (moving via a random walk) before creating a new edge on the graph (a,c).

A wide variety of either open source and public domain random graph libraries can be found. The Cytoscape graph visualization package contains a number of ‘add-on’ libraries written in the Oracle™ JAVA™ programming language. The models include those of Gilbert and Erdos and Renyi discussed here. The Graph-Stream open source project includes a number of random graph generators. GTgraph was written for the DIMACS 9th challenge on shortest path analysis and includes small world and Erdos-Reyni model. The Networxx library contains a large number of methods in addition to those detailed in this section. The library has the ability to generate over 80 different graph types. The SPRAND library detailed in Cherkassky *et al* (1996) has been used widely in a number of seminal works into the application of single criteria shortest. The general approach of SPRAND model is given in Algorithm 2.4. A simplified view is presented where no error checking based on user input or user specified weight ranges are provided. The algorithm outlines the basic approach where the user specifies just the number of edges and vertices present in the graph together with a seed value for the random number generator.

Algorithm:	SPRAND Random Graph Generator
Input:	N = The Number Of Nodes In The Graph M = The Number Of Edges Of The Graph $SEED$ = The Seed Value For A Random Number Generator
Output:	A Series Of String Detailing Edges In Graph In The Format: a source target weight
Initialize a random number generator with $SEED$ value Generate a edge from 1 To 2 with a random weight Generate a edge from N to 1 with a random weight $EDGES = 2$ FOR ($I = 2$ TO $N-1$) { IF ($EDGES == M$) Break; // Exit The For Loop ELSE { Generate a edge from I To $I+1$ with a random weight $EDGES++$ } } WHILE ($EDGES < M$) { E = Generate a new edge between random vertices with random weight WHILE (E Is Duplicate Of Existing) E = Generate a new edge between random vertices with random weight $EDGES++$ } }	
Algorithm 2.4 SPRAND Random Graph Generator Model	

2.4. Single Criteria Path Optimisation

Shortest path analysis on graphs has been studied for many years with notable and widely used algorithms being seen in the form of those of Dijkstra (1959), Bellman-Ford (1957) and Johnson (1977). As already indicated Pallotino and Scutella (1998) highlight the large research effort that has been undertaken on the single criteria shortest path problem. Figure 2.17 presents the results returned from a ScienceDirect key word search containing 'Shortest AND Path AND Analysis'. Since the turn of the century a further 1,618 papers have been published on the subject.

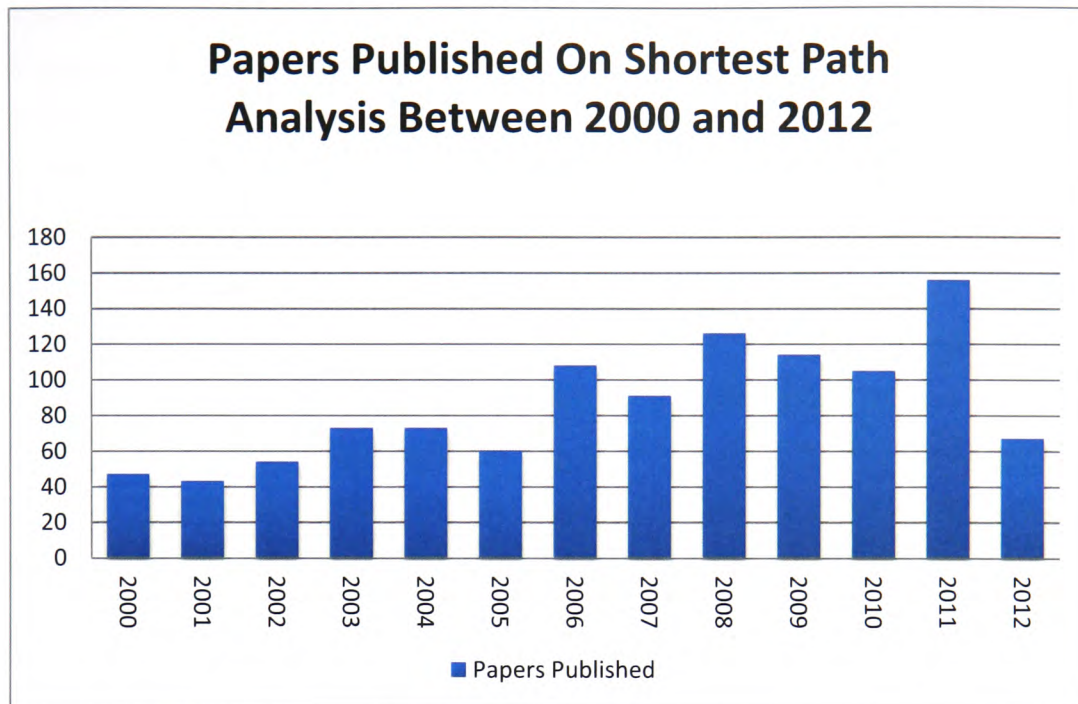


Figure 2.17 Papers on Shortest Path Analysis Published Between 2000-2012

The shortest path problem is one of the most widely studied graph and network optimisation problems. It has attracted researchers from a wide variety of fields and disciplines. These fields include, but are not limited to, transportation, logistics, computer networking and the utilities. Managbanag *et al* (2008) presents a high-level study into the use of shortest path analysis to determine longevity further indicating the widespread and cross-disciplinary adoption of graph optimisation theory. Cherkassky *et al* (1994) argue that despite the amount of published literature advances in the field of shortest path analysis continue to be made. It should be noted however that those advances come not in improving the effectiveness of the various algorithms which have been proven correct, but rather the efficiency with which the calculation is performed. In Cormen *et al* (2001, p.597) it is stated “The algorithm does indeed compute the shortest path” when discussing the Dijkstra shortest path algorithm.

Several complete and comprehensive studies of the shortest path problem can be found in the literature. Notable examples of these can be seen in Ahuja *et al* (1993) and the previously cited study by Pallotino and Scutella (1997). Zhan and Noon (2000)

present a study into the use of shortest path algorithms on real world road networks. Zhan (1997) argues that many of the studies of the shortest path problem on real road networks will consist of networks made up of perhaps several thousand vertices. The study undertaken by Cherkassky *et al* (1994) involved the development of a library of both random graph generators (SPRAND) and shortest path analysis algorithms (SPLIB). These codes provided the basis of the graphs analysed by the studies of shortest path problems undertaken by Pallotino and Scutella (1997). The works of Zhan and Noon (2000), Cormen *et al* (2001) and Ahuja *et al* (1993) present a series of typologies for the style of problems that can be solved using shortest path algorithms. The nature of graph-based problems can be seen in the following typologies: a) the one to all problem; b) the one to some problem; c) the one to one problem.

The outputs from the studies of Pallotino and Scutella (1997) and Cherkassky *et al* (1994) highlight that no algorithm can be said to suit all kinds of problems under all circumstances. There is no 'best' single criteria shortest path algorithm and-that the performance of any algorithm is likely to depend on the size of the network being studied together with the data structure used by the algorithms. Jacob *et al* (1999) take this argument further stating that the connectivity associated with real world graphs tends to be lower than that of some of the graphs studied as part of other reviews. Jacob *et al* (1999) argue that a density ratio of around 2.6 is more likely to be seen in the real world. They compare those levels of density with those analysed in the work of Pallotino and Scutella (1997) who use an edge/vertex ratio up to 10. The authors also highlight that real world graphs are likely to differ in their structure (as in Figure 2.1) from their randomly generated counterparts. The general view that there is no singularly optimal algorithm is agreed upon by Cherkassky *et al* (1994, p.516) who state "Although our research does not produce a single best code for the shortest path problem, two codes we developed are very competitive in their domains". The authors report that the Dijkstra algorithm with double buckets produced high quality results on graphs with non-negative edge lengths with the topological algorithm of Goldberg and Radzik (1993) being optimal of graphs where negative edge costs are present.

In order to ensure brevity and succinctness the various algorithms covered are not in detail although key observations are made. It should be noted that given the ‘classical’ nature of the problem the majority of text books reviewing data structures include a section on shortest path problems, notably Dijkstra’s algorithm (Weiss 1994, Rosenstein 1988 and Melhorn and Saunders 2008). Shortest path algorithms can be separated into two distinct but related groups, the label-correcting algorithms and the label-setting algorithms. The key differences between the two formations relates to how the algorithm converges towards the optimal solution. Label-setting algorithms can terminate when the destination vertex has been scanned and finalized. Label correcting must continue until all vertices making up the graph have been scanned and finalized. The Dijkstra shortest path algorithm is probably the most widely considered single criterion shortest path algorithm and can be seen to be a member of the label setting family. The algorithm itself is outlined below.

The Dijkstra approach to the solution of shortest path problems is unable to handle edges where there may be a negative cost or weight. It is commonly accepted that the performance of the algorithm is limited in the above Algorithm 2.5 in the line ‘*Vertex $v = \text{Element From } L \text{ With Shortest Distance}$* ’ where $O(|V|^2)$ in line comparisons must be made. More advanced data structures and mechanisms may be used to vastly improve the performance of the algorithm. Binary heaps (Figure 2.18) have been shown to be successful in accomplishing this task.

The binary heap maintains an ordered structure where the minimum vertex is always the root. Use of the structure therefore leads to improvements in the runtime of Dijkstra’s algorithm resulting in a runtime of $O(|E| * \log |V|)$. Fredman and Tarjan (1987 p.597) introduce the Fibonacci heap suggesting “For situations where the number of deletions is small in relation to the total number of operations, F-Heaps are asymptotically faster than binominal queues”. Kingston (1998) suggests that the Fibonacci heap is the most efficient implementation of the full complement of priority queue operations that is currently known. Figure 2.19 (Ahuja, 1993) presents a view of Fibonacci heap while Figure 2.20 demonstrates the complexity introduced with the structure.

Algorithm:	Dijkstra Shortest Path Algorithm
Input:	S = Source Point Of Shortest Path T = Target Point Of Shortest Path $G = (V , E)$ – A Graph Representing The Network Topology $C[E]$ = Costs Associated With Each Graph Edge
Output:	Shortest Path Tree From S
$D[V]$ // Array Storing Shortest Distances From S $P[V]$ // Array Storing Predecessor Information For Each Node L // List Of Unsettled Nodes $D[S] = \infty$ $P[S] = 0$ $L = L + S$ WHILE L NOT EMPTY { Vertex v = Element From L With Shortest Distance Remove v From L IF ($D[v] == \infty$) Break; FOREACH neighbour u OF v { $TEMP = D[u] + C[v, u]$ IF ($TEMP < D[u]$) { $D[u] = TEMP$ $P[u] = v$ $L = L + u$ } } } }	

Algorithm 2.5 Dijkstra's Shortest Path Algorithm

Cormen *et al* (2001) highlight the relative complexity of the Fibonacci heap raising a question as to whether the complexity involved in the implementation of the algorithm outweighs the benefits of using it. Table 2.2 presents an analysis of the run time of various binary and Fibonacci heap operations. It should be noted however that Ahuja *et al* (1993) suggest that the Fibonacci heap was optimal on the networks used in their study. Dial *et al* (1979) introduce the use of the Single Bucket structure to the Dijkstra (1959) shortest path algorithm. In Goldberg and Silverstein (1995) a

comprehensive study into the use of buckets to improve the efficiency of the Dijkstra shortest path algorithm is performed. The bucket structure is further considered in Cherkassky *et al* (1994) where a comparison of buckets and other structures, including the previously discussed heaps is undertaken. Figure 2.21 represents the single bucket structure taken from the implementation of Dial *et al* (1979).

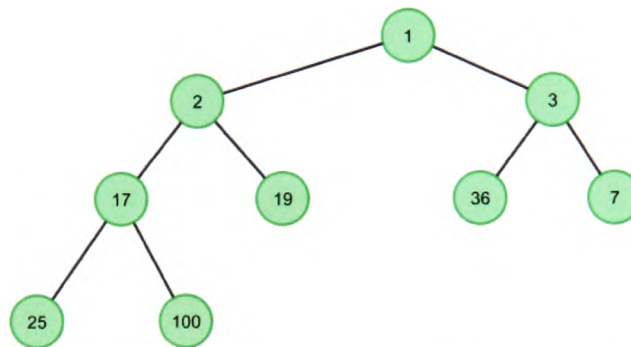


Figure 2.18 Basic Binary Heap Structure (Cormen et al, 2001)



Figure 2.19 Basic Fibonacci Heap Structure (Ahuja et al, 1993)

I	1	2	3	4	5	6	7	8	9	10	11	12	13
Pred(i)	0	0	2	2	0	5	0	2	5	7	4	3	3
SUCC(i)	∞	3,8,4	13,12	11	6,9	∞	0	∞	∞	∞	∞	∞	∞

Figure 2.20 Organisation of Heap in Figure 2.19

Procedure	Binary Heap	Fibonacci Heap
Make Heap	$\Theta(1)$	$\Theta(1)$
Insert	$\Theta(\log V)$	$\Theta(1)$
Minimum	$\Theta(1)$	$\Theta(1)$
Extract-Min	$\Theta(\log V)$	$O(\log V)$
Union	$\Theta(V)$	$\Theta(1)$
Decrease	$\Theta(\log V)$	$\Theta(1)$
Delete	$\Theta(\log V)$	$\Theta(\log V)$

Table 2.2 Heap Operation Times (Cormen et al, 2001)

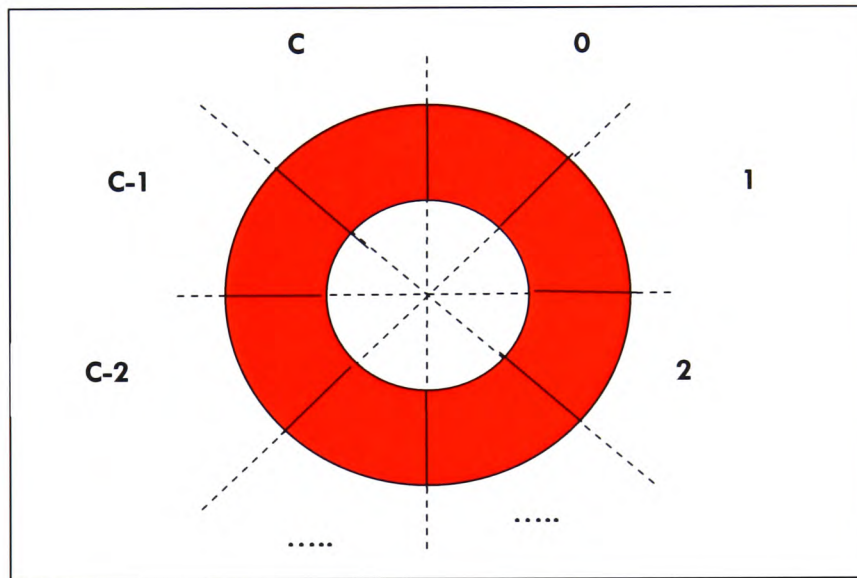


Figure 2.21 Single Bucket Structure (Dial, 1979)

In Dial's implementation an array of 'buckets' is maintained with the i^{th} bucket containing all vertices v where $d(v) = i$. When the distance label of a vertex is updated it is removed from its old 'bucket' and inserted into the bucket corresponding to the updated distance value. The implementation maintains an index M . Initially $M = 0$ and has the property that $i < M$ buckets are empty. The next vertex to be scanned is removed from the bucket M , or if the bucket is empty, M is incremented. One of the major issues associated with the use of the single bucket as a data structure can be seen in the memory requirements of the algorithm when dealing with large graph structures consisting of potentially millions of vertices and edges with a large range of edge

lengths. Suggestions to remedy the memory requirements issue are given in Cherkassky *et al* (1994). The authors make use of an ‘overflow’ bag to reduce the amount of memory occupied by the main structure.

The empirical study undertaken by Cherkassky *et al* (1994) concluded that the results obtained from the implementation of the Dijkstra algorithm in conjunction with a double bucket structure were optimal stating that the implementation was “best in most situations”. Zhan and Noon (2000) in their study of the shortest path problem on real roads confirm these results. A total of D buckets in the low-level bucket set are used. A bucket i in the high-level buckets contains all vertices whose distance labels are within the range of $\{i*d, (i+1)*d-1\}$. In addition, a non-empty bucket with the smallest index L is also maintained in the high-level buckets. A low-level bucket $d(j)-L*d$ maintains vertices whose distance labels are within the range of $\{L*d, (L+1)*d-1\}$. Vertices in the low-level buckets are examined during the scanning process. After all vertices in the low-level buckets are scanned the value of L is increased. When the value of L increases all vertices in the nonempty high-level buckets are moved to its corresponding low-level buckets and the next cycle of the scanning process begins (Cherkassky *et al*, 1994). In order to retrieve the distance associated with a given vertex from the graph the algorithm first finds the top-level bucket associated with that vertex. In Cherkassky *et al* (1994) they conclude that Dijkstra’s algorithm with double buckets provided optimal performance on all problems except in the case of small grid graphs. Zhan (1997) undertakes a comprehensive study into shortest path problems on real world graphs implementing tests with a variety of shortest path algorithms and data structures. The runtimes suggested in that study are reproduced in Figure 2.22.

Abbreviation	Implementation Description	Complexity
BF	Bellman-Ford-Moore basic implementation	$O(V , E)$
BFP	Bellman-Ford-Moore with parent-checking	$O(V , E)$
DIKQ	Dijkstra's Native implementation	$O(V ^2)$
DIKB	Dijkstra's using buckets structure – basic implementation	$O(V ^2)$
DIKBM	Dijkstra's using buckets structure – with overflow bag	$O(E + V C)$
DIKBA	Dijkstra's using buckets structure – with approximate buckets	$O(E + V (C/A + A))$
DIKBD	Dijkstra's using buckets structure – double buckets	$O(E B + V (B + C/B))$
DIKF	Dijkstra's using heap structure – Fibonacci heap	$O(E + V \log(V))$
DIKH	Dijkstra's using heap structure – k-array heap	$O(E \log(V))$
DIKR	Dijkstra's using heap structure – R-heap	$O(E + V \log(C))$
PAPE	Incremental Graph – Pape-Lewit implementation	$O(V ^{2V})$
TWO.Q	Incremental Graph – Pallotino implementation	$O(V ^2 E)$
THRESH	Threshold Algorithm	$O(V , E)$
GOR	Topological Ordering – basic implementation	$O(V , E)$
GOR1	Topological Ordering – with distance updates	$O(V , E)$
Notation: V is the number of network vertices. E is the number of network arcs C is the maximum arc length in the network A and B are input parameters		

Figure 2.22 Runtime of Shortest Path Algorithms (Zhan, 1997)

2.4.1. Alternative Shortest Path Algorithms

The Bellman-Ford algorithm (1958) does not require positive edge values. The algorithm is in general structure very similar to that of the Dijkstra algorithm. However, instead of selecting the connected vertex with the minimum value (*'Vertex $v = Element From L With Shortest Distance$ '*) the algorithm performs the relaxation process on all connected edges. The iteration of the process cause this to be performed $|V| - 1$ times where $|V|$ is the number of vertices in the graph. Despite being able to handle negative edges the algorithm fails to handle scenarios where negative cycles may be found. With regard to negative edge capability of the Bellman-Ford algorithm, the Johnson algorithm (1977) also allows edges with negative weights. The algorithm makes use of both the Bellman-Ford algorithm to handle scenarios where there may be negative edges before applying the Dijkstra shortest path algorithm. The Floyd-Warshall algorithm (1962) computes the shortest paths between all pairs of vertices on a graph and is able to do so with the worse case performance of $\Theta(|V|^3)$ where $|V|$ is the number of vertices in the graph. In un-extended form the algorithm only returns lengths of the path and requires a modification, albeit a simple one to return the paths themselves.

The A* algorithm (Hart *et al*, 1968) is widely studied in Artificial Intelligence (AI). Originally conceived as an extension to the Dijkstra algorithm it is suited to real world problems through the use of distance based heuristics. When applied to the shortest path problem the term heuristic describes the introduction of a function to estimate the distance from the current vertex i to the destination vertex. Within algorithms such as the A* algorithms the heuristic typically used can be identified as:

$$f(i) = (g(i) + h(i))$$

Equation 2.4 A* Heuristic Mechanism

Where g is the distance travelled from the source vertex to the current vertex i . The value $h(i)$ is an estimated (heuristic) value from the current vertex to the end vertex. In a GIS based systems where the actual coordinates of the current position and the end point are known this can be as simple as:

$$h(i) = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

Equation 2.5 Distance Calculation

Where $\{x1, y1\}$ and $\{x2, y2\}$ each represents a two dimension location coordinate.

In Jacob *et al* (1999) a study based on a comparison between the Dijkstra algorithm with a number of data structures is undertaken together with the A* algorithm and an extended A*. The study is devoted to a transport simulation modeller. The heuristic nature of the A* results in the algorithm being featured in various texts on artificial intelligence including Luger (2002) and Russell and Norvig (2003). Although used to identify the optimal path(s) through a graph these texts introduce the algorithm in the context of games and game theory. It should be noted that the A* algorithm will often return near optimal results unlike the Dijkstra algorithm which has been proven to be correct as highlighted in Cormen *et al* (2001). Deckter and Pearl (1995 p. 505) find the optimality of the algorithm dependent on the heuristic function used stating “if the performance tests are confirmed to cases in which the estimates are also consistent, then A* is indeed optimal”.

The current section has sought to introduce the basic concepts of shortest path analysis and basic methods used to introduce performance increases to the problem. The traditional methods used to increase performance of point to point queries are often no longer suitable for the scale of graphs used in the real world, such as continent wide

road networks or large scale social networks. Gubichev *et al* (2010) highlight that a straightforward implementation of the Dijkstra algorithm will take more than 500 seconds on a graphs sized 3,000,000 X 220,000,000. Sections 2.5 and 2.6 introduce state of the art methods used in shortest path analysis and social network analysis.

2.4.2. K Shortest Paths

The K shortest path problem can be considered a generalisation of the more traditional shortest path problem where not only the single shortest path must be returned but several (K) shortest paths. Eppstein (1997, p.2) states that “The K shortest paths problem, for a given K and a given source-destination pair in a digraph, is to list the K paths in the digraph with minimum total length”. The problem was originally examined in Hoffman and Pavley (1959). In Epstein’s methodology, generally repeated in other works, the Dijkstra shortest path algorithm is used on a reverse of the graph, i.e. the shortest path from the destination to every other vertex is calculated and paths of increasing length derived from the resulting shortest path tree. Eppstein (1997) cites many reasons where the calculation of the K shortest paths may be preferable to the calculation of a single path such as problems with associated additional constraints that may be difficult to define. Examples are provided in problems such as the routing of power cables into areas or communities where there may be opposition to their installation for example. Aldious and Fill (1999) suggest the method may be useful in enumerating a series of paths across a graph, as is random walking. Davies and Lingras (2003) also highlight the feasibility of using the K shortest path as a means of generating possible paths for the Genetic Algorithm approach.

The K Shortest path problem may be further categorized based on the plausibility of loops on a path. The introduction to this work suggested that people prefer to travel along paths which they consider to be short, quick but also simple. They may prefer a slightly longer trip (both in terms of speed and time) provided that the journey has some added benefit such as requiring less turns or avoids certain features etc. However, it almost goes without saying that the development of loops within a path

makes little or no sense. Exceptions to this can be seen in scenarios where path turns are prohibited such as ‘No left turns’ etc. The calculation of K paths where loops are not allowed feature in the works of Chen (1994), Yen (1971), Lawler (1972) and Lawler (1977). The requirement for solution paths to remain ‘perfect’ or ‘loopless’ makes the problem considerably more difficult to compute (Eppstein, 1997). The Yen (1971) implementation of the problem features a runtime of $O(k|V|(|E|+|V| \log |V|))$. Yen (1972) presents a variation of his earlier version of the K shortest path algorithm which introduces performance increases. Further increases to the Yen approach are presented in Pascoal and Martins (2003) where the runtime is reported as $O(k|V|(|E|+|V| \log |V|))$. Despite having the same theoretical worse bounds as the original implementation of Yen the Pascoal and Martins’ approach in practice demonstrates substantial increases in performance. Pascoal and Martins make use of a Fibonacci structure to sort candidate paths and generate deviation paths from the destination node. Yen (1971 and 1972) operate in the reverse generating candidate paths from the source. Brander and Sinclair (1995) present a comparative analysis of K shortest path algorithms. In their (Brander and Sinclair) study the authors attempt to identify bottlenecks in the algorithm notably highlighting additional calls to a shortest path algorithm (Dijkstra’s algorithm) in order to derive successive paths in the Yen (1972) algorithm. Brander and Sinclair suggest a number of ‘acceleration mechanisms’ to increase the performance of the algorithm. The methodologies are similar to those of Pascoal and Martins who highlight a 300% increase in performance over the Yen approach across a series of graphs using the procedure highlighted above.

Fox (1975) presents an implementation of the K shortest path problem. Eppstein (1997 p.3) suggests that the algorithm was the best “previously known k -shortest-paths algorithm”. In Eppsteins’ work results are obtained which improve upon those of Fox. Where the Fox algorithm is able to perform in $O(|E| + k|V| \log |V|)$ time the methodology proposed by Eppstein reduces this to $O(|E| + |V| \log |V| + k)$. Hersberger *et al* (2007) present a solution that makes use of the Fibonacci heap structure alongside Dijkstra’s algorithm. The algorithm is tested on real road and random networks. In the case of the real road networks they employ relatively low-resolution road networks consisting of around 5 - 6000 vertices and 12 - 15,000 edges. They report improved results over the Yen (1971) algorithm although the extents of which vary dependent on

the geographical spread of the source and destinations. The methodology makes use of a path replacement strategy which it is suggested operates faster than traditional deviation methods although it is prone to failure. Where failure occurs and is detected a traditional deviation mechanism is performed. The authors report that failure occurs rarely and where it does performance is no slower than that of Yen.

Santos (2006) introduces a K shortest path algorithm which builds on the algorithms presented in Eppstein (1997). The algorithm maintains the path deviation approach of Eppstein but introduces edge cost sorting and effective data structures. The authors report that their algorithm can generate 1,000,000 paths in less than three seconds on random graphs and less than 10 seconds in real world graphs based on the US road network. Hamed (2010) presents a Genetic Algorithm solution to a subset of the K shortest path problem using a novel chromosome representation. The aim of Hameds work is to generate K paths of increasing length in terms of the number of vertices making up those paths. However, the graphs employed in the tests are of limited size ranging from 9 vertices and 13 edges to 20 vertices (edges unknown). In addition the author does not make any comparisons with existing algorithms and thus it is difficult to make any real comparisons with other algorithms.

2.5. State of the Art in Single Criteria Shortest Path Analysis

In the following section a series of methods used to optimize the run time of shortest path queries on large graphs (defined as consisting of several million vertices and edges such as continent sized road networks) are considered. As previously highlighted the Dijkstra shortest path algorithm can solve shortest path queries between two nodes in sub linear time where the algorithm terminates when the target vertex has been scanned. However on very large graphs consisting of several million vertices and edges the application of advanced data structures is not in itself enough to enable the performance of real time queries. Methods such as heap structures or bucket structures will increase the performance of the algorithms but the performance of the algorithms

will remain unsatisfactory for either large numbers of queries or real time systems. Recent work has seen road hierarchies, pre-processing and bi-directional searching as detailed in works such as Möhring *et al* (2005), Möhring *et al* (2007), Hilger *et al* (2006), Geisberger (2008) and Bauer and Delling (2010) and Efentakis *et al* (2011) used to increase the perform of shortest path analysis. The use of a *combination* of such techniques has been shown (Holzer *et al*, 2005; Abraham *et al*, 2011) to reduce the run time of shortest path queries to levels that enable real time analysis of shortest path queries to be performed on large-scale graphs. Here a brief review of these techniques is undertaken.

2.5.1.1 Bi-directional Searching

The bi-directional search is reasonably straightforward extension of the standard shortest path approach. Assuming that two vertices are selected from a graph and referred to as the source (s) and the target (t) of the query the bidirectional approach scans forward from vertex s (to t) and backwards from vertex t (to s). The algorithm requires the generation of a reverse graph from vertex t . The generation of the reverse graph is not however a difficult proposition with modern computing techniques such as object orientation and the use of such techniques is performed in various other graph algorithms including a number of approaches to the K-shortest path problem. The bidirectional approach alternates between the two graphs (forward and reverse) with an iteration performed in each direction. Let $df(u)$ be the distance labels of the forward search and $db(u)$ the labels of the backward search respectively of a given vertex (u) The algorithm can be allowed to terminate when one vertex has been designated to be finalised by both forward and reverse searches. Then the length of shortest path is determined by the vertex u with minimum value $df(u) + db(u)$ and the path itself can be composed from the concatenation of the shortest path from the start vertex s to u (found by the forward search) and the shortest path from u to the destination t (found by the reverse search). Goldberg and Harrelson (2005) introduce a similar bidirectional mechanism to the A* algorithm together with the introduction of landmarks. The ALT* (A*, Landmark and Triangulation) method pre-computes the shortest distance from

each node to a selection of vertices considered landmarks. Landmarks are generated either randomly or through some other more complex selection method. The landmarks selected are then used as the basis for the heuristic functionality of the A* algorithm. Consider the simple graph presented in Figure 2.23. Assume that the vertex labelled v is assigned landmark status - that is to say, the shortest path between it and every other node has been calculated during an earlier pre-processing phase. An accurate heuristic 'distance' can be gathered using the following equation:

$$dist(l, u) \geq (dist(v, u) - dist(v, l))$$

Equation 2.6 ALT* Heuristic Equation

The algorithm operates bi-directionally and as is the case with the Dijkstra algorithm terminates when the target node has been scanned and finalized.

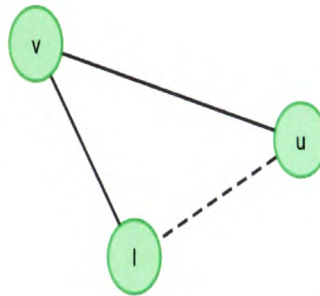


Figure 2.23 ALT* Triangulation Graph

2.5.1.2 Hierarchical Methods

Sanders and Schultz (2005) introduce the notion of highway hierarchies to decrease the runtime of point-to-point shortest path queries through an extension of a multi-level graph model first introduced in Schultz *et al* (2002). The basic idea of the highway hierarchies approach is that outside some local areas (the vertex radius) around the source and the target vertices only a limited subset of 'important' edges have to be considered to find the shortest path. A pre-processing step is performed at which additional edges representing shortest paths between certain vertices are added to the

graph. The additional edges can be seen as shortcuts for Dijkstra's algorithm. These additional edges act as gateways to higher graph levels that coarsen the graph. To find a shortest path between two nodes s and t using a hierarchy it is possible for Dijkstra's algorithm to consider a relatively small sub-graph of the hierarchical graph. The hierarchical structure entails that a shortest path from s to t can be represented by a certain set of upward and of downward edges and a set of level edges passing at a given higher level.

Geisberger *et al* (2008) introduce contraction hierarchies, an approach that like that of Sanders and Schultz (2005) introduces shortcuts between two vertices. The shortcut operation deletes (temporarily) a vertex v from the graph; then for any neighbours of v where $(u, v) \cdot (v, z)$ is the only shortest path between u and z it adds a shortcut edge (u, z) with the weights $w(u, z) = w(u, v) + w(v, z)$ thus preserving the shortest path information. To demonstrate the principle of shortcutting in practice Figure 2.24 and Figure 2.25 are introduced. Figure 2.24 is a simple network of four nodes. Figure 2.25 shows the removal of node four and the introduction of shortcuts with the updated weight information as a result of the mechanism.

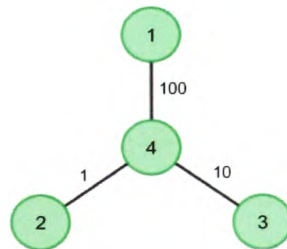


Figure 2.24 Simple Test Graph for Hierarchies

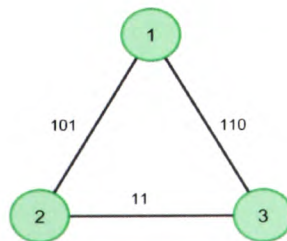


Figure 2.25 Simple Test Graph from Figure 2.24 After Shortcut Mechanism

A contraction hierarchy pre-processing routine defines a total order among the vertices and shortcuts them sequentially in this order until a single vertex remains. Algorithm 2.6 presents a simplified view of the method employed to construct a contraction hierarchy. The algorithm is passed the graph to be contracted together with a priority queue giving the order in which nodes are to be processed. Geisberger (2008) highlights that the order in which vertices are processed may have a substantial effect on the runtime of the pre-processing methodology and perhaps more importantly the resulting queries. Geisberger presents two models with regard to the ordering of nodes:

The aggressive variant

The main consideration is the performance of the queries, with little regard made to the time spent generating the contraction hierarchy.

The economical variant

The product of the average query time and the amount of time spent pre-processing the contraction hierarchy

The selection of various control parameters can be used to manipulate the performance of hierarchy construction between the two variants. The various control parameters form the basis of heuristics to estimate how "important" each vertex is based on local graph properties (such as the number of shortcuts added if the vertex were contracted).

Algorithm:	Simplified Contraction Hierarchy Construction
Input:	A Populated Graph $G=(V , E)$ P = A Priority Queue Of Nodes
Output:	A Modified Graph $GMod=(V , EMod)$
<pre> FOREACH ($u \in V$ ordered by P) { FOREACH ($e(v,u) \in E$ with $v > u$) { FOREACH ($e(u,w) \in E$ with $w > u$) { IF $\langle u, v, w \rangle$ { $EMod = EMod \cup \{u, w\}$ $Cost(u,w) = Cost(u,v) + Cost(v,w)$ } } } } </pre>	

Algorithm 2.6 Simplified Contraction Hierarchy Construction

Following the construction of the contraction hierarchy point-to-point queries are performed on the modified graph structure using a bi-directional variant of Dijkstra's shortest path algorithm. Goldberg (2011) reports exceptional increases in performance seen in the runtime of point-to-point queries using both highway and contraction hierarchies. Table 2.3 provides the average runtime seen on a graph of the Western European road network, a graph consisting of 18 million vertices and 42 million edges with the costs based around travel time as reported in Goldberg (2011). It should be noted that the times given in table report only the time taken to calculate the shortest path. The retrieval of the complete path requires the use of a recursive "unpacking" procedure. The unpacking procedure may take longer than the identification of the shortest distance itself. This is true of both the highway and contraction hierarchies. The tests performed in Table 2.3 utilised a high-end server hence extremely low runtime for the Dijkstra approach relative to the graph size.

Algorithm	Dijkstra	Highway Hierarchy	Contraction Hierarchy
Runtime (Seconds)	2.008	0.0004	0.000094

Table 2.3 Comparison of Hierarchy Based Approaches

Delling *et al* (2011a) introduce the PHAST algorithm (*parallel hardware-accelerated shortest path trees*) which makes use of the additional computational power available via the use of multi-core technology and/or GPU (graphical processing unit) processing power to increase the performance levels seen in calculating shortest paths using contraction hierarchies. Of particular interest in the work is authors' suggestion that because of hierarchal and related structures the speed at which the performance of shortest path queries are performed is no longer limited by the power of the CPU but rather the memory bandwidth available. By moving to GPU based searches the results produced in Delling *et al* (2011a) indicate that despite the lower processing power of GPUs the ability to process multiple vertices in parallel together with increased memory bandwidth available can lead to extreme performance increases. Before discussing the runtime reported for the PHAST algorithm it is worth highlighting that the results reported in Table 2.3 relate to the analysis of point-to-point queries whereas the PHAST algorithm aims to produce the shortest path tree - that is to say the shortest path from a single node to all other nodes. The authors report that when using a GPU based technique they were able to generate 16 shortest path trees simultaneously with an average runtime of 2.2ms. Finally the authors report being able to produce an all-pairs shortest path analysis on Western European roads in a little over 11 hours.

Abraham *et al* (2010) introduce an algorithm based upon contraction hierarchies that has particular relevance to the work conducted in this thesis. The authors highlight the multi-criteria nature of path planning and aim to generate alternative paths (to the shortest). The graph is firstly pre-processed using contraction hierarchies and the shortest path between two nodes acquired using the "unpacking" methodologies previously highlighted. The authors then attempt to generate an alternative path with the properties that the similarity (number of vertices shared) must be low and that the path

must be ‘reasonable’ with a limited number of unnecessary detours. In order to find alternative paths the authors use a principle of local optimality of sub paths where each sub path will also be the shortest path between the source and destinations of the sub path. Therefore the algorithm requires multiple shortest path calls, the number of which will increase as the number of nodes making up the shortest path increases. The authors find report promising results based on speed with the algorithm finding one alternative path in 3.1ms and three alternative paths in 3.9ms using contraction hierarchies. The success rate of the algorithm using the contraction hierarchies is limited with the algorithm returning a valid alternative path in only 58% of cases. A bi-directional variant of Dijkstra’s algorithm is seen to have a much higher success rate (94% for one alternative, 62% for three alternatives) but completes a processing run much more slowly at between 26 and 33 seconds dependant on the number of alternative paths sought between one and three.

2.5.1.3 Transit Node Approach

Bast *et al* (2006) introduce the transit node methodology. The method takes advantage of the small world phenomenon that exists in real world road networks. That is to say for any given region there exists a small number of vertices such that any shortest path to a distant vertex will pass through a member of that small number of vertices. The algorithm segregates the graph into regions (based upon a regular grid) at multiple levels. Bast *et al* (2006) find that the US road network can be adequately separated into two grid levels with the first level consisting of a grid of 128x128 and a lower level of 256x256. For the sake of clarity Figure 2.26 is introduced. It consists of an upper level grid (*OUTER*) of size 9x9. *INNER* is a lower level grid sized 5X5 that is placed within *OUTER*. Cell *C* occupies the centre of *INNER* in this example although during processing cell *C* will move around the grid *INNER* assisting the calculation of transmit nodes. The edge set E_C consists of those edges that have only a single vertex within cell *C*. For the purposes of the transit node approach edges where both vertices are within cell *C* are considered a local search and handled separately to the transit node methodology. The set V_C refers to those vertices in the cell *C* that are present in set E_C .

The set of transit nodes for the cell C is the set of vertices v from V_{INNER} with the property that there exists a shortest path from some vertex in V_C to some vertex in V_{OUTER} that passes through v . In order to compute the set of transit vertices for C it is required to compute all-pairs shortest paths between V_C and V_{OUTER} , marking all vertices present in V_{INNER} as being transit nodes. An iterative process considers each cell C in $INNER$, with the union of all transit vertices for each and every cell being considered the set of transit nodes for the entire graph. The use of the grid structure reduces the number of all-pairs shortest path calculations required. Bast *et al* (2006) find that in the US road network with the grid sized previously highlighted (128 x 128 and 256 x 256) there are around 8,000 transit nodes. Following the identification of each transit vertex in a graph a further all-pairs shortest path analysis is carried out between each transit vertex. The production of the shortest path between any two non-local vertices is then the relatively straightforward task of concatenating a series of pre-calculated shortest path, as shown in Algorithm 2.7 .

Where both the source and target are considered local the shortest path is identified using standard shortest path analysis techniques such as contraction hierarchies. For non-local searches the calculation of the shortest path becomes a series of look-up table searches as all the information required has been pre-calculated (the distance of each vertex to its nearest transit node and the distance between each transit node). The authors report a longer pre-computation time (around 20 hours) but an exceptional level of performance when carrying out point-to-point queries. Searches complete in around 1.2 microseconds for distant searches and 5112 microseconds for local searches on the US road network. For local searches the authors make use of highway hierarchies but other techniques such as contraction hierarchies are legitimate. In addition using alternative methods would decrease both the pre-computation phase and speed at which local searches are performed as seen in Table 2.3 if comparing highway hierarchies to contraction hierarchies. As with other hierarchy based methods determining the shortest path travelled increases the processing time required with Bast *et al* (2006) reporting an increase from 1.2 microseconds to 5 microseconds for full path retrieval with non local searches.

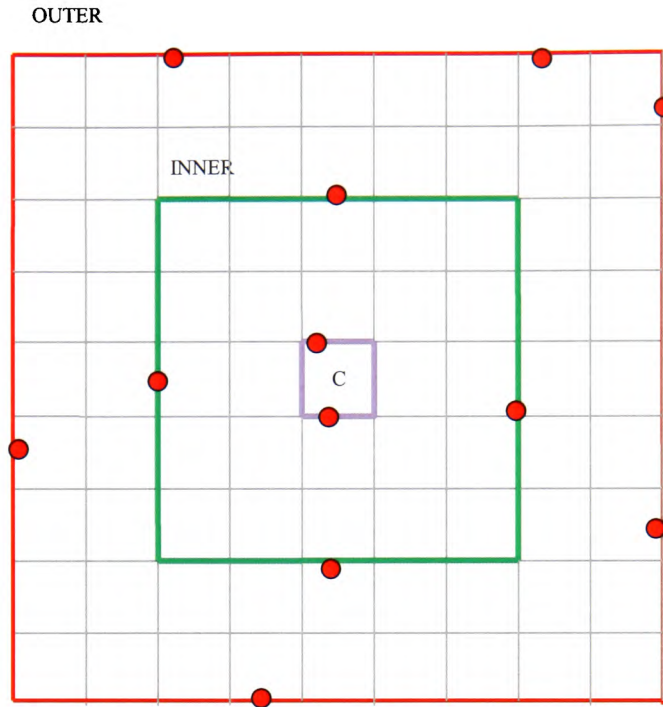


Figure 2.26 Example grid model of Bast et al (2006)

Abraham *et al* (2011) introduce a hub labelling approach. The technique shares certain features with that of Bast *et al* (2006) in that both reduce the shortest path problem to a series of table lookups. In the Abraham *et al* (2011) methodology a graph is subjected to a pre-processing mechanism using the contraction hierarchies approach. The work stores a large amount of information (the shortest paths to other vertices) within a label structure associated with a vertex. Each vertex directly provides the details of shortest paths to around 85 other vertices following pruning. The authors report being able to retrieve least cost distance between any two vertices on continental sized graphs in 276 nanoseconds on a graph of Western Europe roads and 266 nanoseconds on a graph of US roads. The work is sparse on details hence the limited discussion here. The authors highlight that in many cases they are able to retrieve the shortest distance in five clock cycles. The fast performance comes with the cost of higher storage requirements associated with the additional information stored within labels. Retrieval of the full path description would take longer as with other methods. The method is extended in Abraham *et al* (2012) to enable the development of queries

from SQL databases. Following the pre processing stage of Abraham *et al* (2011) the resulting graph structure is stored in a database. The work highlights how the retrieval of not only the shortest path is possible but information that may be related to the shortest path such as points of interest along the route. In Abraham *et al* (2012) the authors report being able to retrieve the full path distance of random point to point queries in 10 - 25ms on the Western European road network (as used in Abraham *et al*, 2011). Retrieving the distance only is possible in an average of 4-5ms. The wide range in average time is seen as a result of the underlying database technology. Where only a limited number of queries are made the system has limited possibilities to take advantage of cache (for SQL queries) memory. As the number of queries increases then more use is made of cache hence the lower average runtime seen when the number of queries is increased.

Algorithm:	Shortest Path Calculation Using Transit Nodes
Input:	S = Source Point Of Shortest Path T = Target Point Of Shortest Path $TSRC$ = Transit Nodes Of Cells Surrounding S $TTRG$ = Transit Nodes Of Cells Surrounding T TN = 2 Dimensional Array of Shortest Paths Between Each Transit Node
Output:	SPD = Shortest Path Distance Between S and T
<pre> IF (!LOCAL_SEARCH) { Ds = Calculate Cost To Nearest Transit Node To S // Pre-computed So Table Lookup Search Dt = Calculate Cost To Nearest Transit Node To T // Pre-computed So Table Lookup Search Dm = Calculate Shortest Path Between $TN[S]$ and $TN[D]$ $SPD = Ds + Dm + Dt$ P = Unpack Path Using Recursive Mechanism } ELSE { P = Shortest Path Computed Using Highway Hierarchies // Or other high performance method } </pre>	

Algorithm 2.7 Calculation of Shortest Path Using the Bast et al (2006) Model

2.5.1.4 Multi Level Pre-processing

Delling *et al* (2011b) describe a methodology for generating ‘customizable’ route plans enabling ‘virtually’ real time cost updates. The method segregates the road network into two distinct groups. The first consists of the road topology and remains relatively static. Graph related aspects related to the topology are the connectivity of the road network and associated attribute information such as road type and speed limit. The second group consists of the metrics used in identifying the shortest path. Delling *et al* suggest that the metrics such as travel time or distance may change frequently or that more than one metric may exist in parallel. A one-to-many relationship exists between the topology of the road (one) and the metrics used in computing the shortest path (many). The methodology proposed by Delling *et al* (2011b) separates the path planning process into three distinct groupings. The first involves the pre-processing of the relatively static road network. This initial stage of the process will exhibit a longer run time. It is however, run infrequently, as the underlying road topology will change infrequently. The phase is characterised by a longer run time with a large amount of additional data (comparable to the size of the graph) being created. The second phase of the process optimizes each cost on the graph (travel time or distance) and operates in a matter of seconds. Their work aims to minimize the amount of pre-processing required when metrics are updated enabling real time traffic congestion information to redirect users to alternative routes.

The methodology makes use of the PUNCH (Partitioning Using Natural Cut Heuristics) method identified in Delling *et al* (2010) that splits the graph into a number of partitions referred to as cells. A cell may be of an irregular geographic shape. Given an input parameter U that represents the maximum size of the cell in terms of the number of edges contained in that cell PUNCH partitions the graph into cells of size at most U while minimizing the number of edges between cells. A shortcutting procedure is then used to assist in the generation of an additional graph hierarchy or level (H). The graph H contains the shortest paths between vertices forming an edge where either endpoint is not in the same cell as the other endpoint. A shortcut edge in H is used to

represent the shortest path for each boundary edge in a cell. The similarities to transit nodes should be clear from the description. To perform a query between s and t , a bidirectional version of Dijkstra's algorithm is performed on the graph consisting of the union of H , C_s (the cell containing the source), and C_t (the cell containing the target). The authors further refine the method through the introduction multiple hierarchies in the graph H . The punch algorithm is applied to the graph with multiple U values (the maximum size of the cell). Determining which vertex sets to be included in the query search is then a case of calculating the graph levels to use based upon table lookups. In order to present the performance of the approach a subset of the results reported in Delling *et al* (2011b) are reproduced below in Table 2.4. The first column gives the method used (MLD- X is the developed algorithm with the graph separated into X number of levels or hierarchies. The second column gives the U size parameters used in the creation of the X levels. The third column gives the amount of time required to pre-process the cost metrics with the space required following the pre-processing phase. The final column gives the query time. The table also reports the information for the two models of contraction hierarchy previously highlighted. The test data is a graph of Western European roads with travel times the cost metric considered. Similar metrics are seen in the full paper (Delling *et al* (2011b) for the other metric considered – distance.

As seen in Table 2.4 the contraction hierarchy is able to perform queries much more quickly than the method developed by Delling *et al*. In contrast, however, processing time for the test data sets is much lower with the proposed algorithm with the additional benefit of a lower storage space requirement. The authors conclude that for the specific application the benefits (processing time, space) seen in the approach outweigh the costs of a reduced runtime. The runtimes seen by the approach are still extremely promising.

Method	U Values	Processing Time (Seconds)	Space (MB)	Query Time (ms)
MLD-1	2^{14}	4.9	10.1	5.81
MLD-2	$2^{12}:2^{18}$	5.0	18.8	1.82
MLD-3	$2^{10}:2^{15}:2^{20}$	5.2	32.7	0.91
MLD-4	$2^8:2^{12}:2^{16}:2^{20}$	4.7	59.5	0.72
CH economical	N/A	178.4	151.3	0.12
CH generous	N/A	355.6	122.8	0.10

Table 2.4 Runtimes Seen in the Delling et al (2011) Approach

2.5.1.5 Practical Implementations

Having discussed the basic methodology of a series of what can be considered ‘state of the art’ methods of computing shortest paths on graphs attention is now turns to how these can be translated to useable applications. Of primary interest would be the web based routing applications provided by Google™ and Microsoft™ Bing™. In the case of Bing™, Microsoft™ has openly stated (Pendleton, 2012) that the methodology employed by the routing algorithm is that of Delling *et al* (2011b) discussed previously. The use of the methodology is relatively recent - prior to the start of 2012, Bing™ maps used a “modified Dijkstra algorithm” (Pendleton, 2012) although the exact nature of the modifications are not disclosed. The only comparison given is that the Delling *et al* (2011b) approach processes queries twice as fast as the modified Dijkstra search it replaced.

Determining the shortest path methodology employed by Google™ Maps is not possible with only scant information being available. Brummit (2007) made brief mention how large amounts of processing (taking 10 months) were required to generate the datasets required for routing. It is *possible* that previously the techniques involved the pre-computation and storage of results with encoding and decoding made possible using the Encoded Polyline Algorithm (Google, 2012). However, the use of such techniques would not realistically enable the introduction of real time traffic information a feature now (though not in 2007) considered an advantage of the Google™ Maps application. An “educated guess” might be that the methodology is based the Transit Node approach of Bast *et al* (2006). Such a guess however would only be based upon the fact that the principle author of that work is employed in a research capacity at Google. Bast *et al* (2010) provide details of mechanisms to increase the performance of public transport search mechanism employed by Google. The method is based around the identification of hub stations, those stations or stops that are used frequently on public transport journeys. The authors however highlight that the shortest path on road networks is inherently different to that of public transport that relies on other factors such as travel time and the intersection of travel times with departure of potential connections.

ESRI (2005) describe the use of Hierarchal structures in their desktop routing solution (ESRI Network Analyst). They highlight that the hierarchal methods employed may return sub optimal paths when compared to the alternative “exact best” method, it is stated “The classical best route algorithm, best known as the shortest path algorithm cannot support real time queries on large network. Sophisticated performance enhancing techniques such as heuristics and hierarchal algorithms dramatically reduce the run time needed to perform the problem”. A number of routing applications or code libraries are available that make use of OpenStreetMap (OSM) information. OpenRouteService (www.openrouteservice.com) is a web based application made available as a web page or a series of web services. The routing methodology is based upon the A* algorithm. Osmsharp (<http://osmsharp.codeplex.com>) is an open source desktop library written in the Microsoft C# language that enables the import and display of OSM data. The library

includes a number of routing algorithms including basic forms of the contraction and highway hierarchy methods reviewed elsewhere in this section. Vetter (2010) developed the MoNav mobile routing application based around OSM data sets. The application makes use of contraction hierarchies to solve routing queries. Vetter highlights a runtime of 273 milliseconds on European datasets. The work is further highlighted in Luxen and Vetter (2011). The pgrouting project (<http://pgrouting.org/>) allows for the import of OpenStreetMap data. The project enables the application of generalized SQL (structured query language) queries to solve routing problem. The routing engine employed is based around the A* and Dijkstra shortest path algorithms.

2.5.1.6 Summary of the State of the Art in Shortest Path Analysis

Recent years have seen a range of high performance methods for decreasing the runtime of shortest path queries on very large-scale graphs. Due to these methods it is possible to perform continent level queries in fractions of second as seen in Abraham *et al* (2011) and Delling *et al* (2011b). The later of these is of particular interest given everyday use of the approach via the Microsoft Bing maps routing application. At the heart of the described improvements are methods such as the use of graph portioning (as seen the described transit node and customizable route planning methods previously described) and use of hierarchal structures such as contraction hierarchies. In certain cases the performance of shortest path analysis is reduced to simply retrieving the path from one or more lookup tables.

2.6. Social Network Analysis

The previous section detailed a series of approaches that have been seen to increase the performance of the Dijkstra shortest path algorithm on large-scale graphs in the form of road networks, with the datasets representing the road networks of the US or Western Europe being dominant in the literature reviewed. Techniques developed in the transportation domain exploit properties of transportation networks such as their low

degree and the presence of a hierarchy based on the importance of roads to achieve performance increases in point to point analysis for shortest paths. In this section alternative large-scale graphs are considered in the form of those seen in social networking.

Graphs seen in social networks vary in size to those seen in real world networks such as roads. Table 2.5 highlights the number of nodes and edges seen in a series of social networks analysed in the literature. The table highlights the network considered, the number of vertices and edges and the density seen in the graph. The final column provides the citation where the graph size is considered. It can be seen from Table 2.5 that the size of the graphs considered is often larger than road networks with a number of sizes presented in the previous section. In addition, social networks exhibit a much higher density than can be seen in road networks. Jacob *et al* (1999) highlights that road networks will often have of a density of around 2.6. Analysis of continent level road networks studied as part of the 9th DIMACS challenge confirm similar density levels with US networks having a directed density of 2.43 and European road networks having a density of 1.21 in undirected form. A loose assumption that the majority of those roads are directed would give a density of around 2.42. The test data sets used for this study and introduced in Chapter 4 have a density of 2.0 – 2.4. There are of course exceptions. The road network of New York is seen to have the highest density of road networks reviewed with a density of 2.78 in a graph sized 264,346 X 733,846.

The notion of what constitutes an edge on a graph appears to affect the average density of the reported graphs. For instance, in some cases an edge is considered the friendship between two individuals. In such cases, as seen in the studies of Table 2.5 considering Skype, Flickr and Twitter the density is, although higher than that seen in road networks much lower than that seen in the studies reviewing Facebook and Orkut where an edge is considered by the authors as a contact between two users such as replying to a message. In such cases the density of the graph is greatly increased.

<i>Social Network</i>	<i>Vertices (Million)</i>	<i>Edges (Million)</i>	<i>Density</i>	<i>Cited In</i>
Orkut	3	220	73.3	Gubichev <i>et al</i> (2010)
Skype	454	3100	6.8	Tretyakov <i>et al</i> (2011)
DBLP	0.77	2.6	3.37	Tretyakov <i>et al</i> (2011)
Twitter	12.9	90.8	7.03	Poblete <i>et al</i> (2011)
Flickr	2.57	33.1	12.87	Mislove <i>et al</i> (2008)
Facebook	0.06	1.5	25	De Meo <i>et al</i> (2012)

Table 2.5 Sizes of Various Social Networks

The levels of density seen in online social networks are generally closer to those seen in works such as that of Pallottino and Scutella (1997) that use a density of up to 10. Therefore the analysis of social networks presents a different set of challenges to those of roads. Riberio and Towlsey (2010) highlight that even the seemingly simple task of acquiring accurate metrics as to the size of online social networks is one that is fraught with difficulties. Whilst social networks will highlight the absolute number of users those networks have (Forbes, 2012) acquiring lower levels of information regarding the network to any degree of certainty is a difficult proposition. In many cases (Flickr, 2012; Twitter, 2012), on-line social networks have public facing Application Programming Interfaces (API) to allow for the querying of the network. Such queries however are often restricted for reasons of network performance. User privacy settings will also often limit the availability of information regarding the network (Krishnamurthy and Wills, 2009). Valafar *et al* (2009) highlight that the concept of friendships in social networks are often not reciprocated with the result that graph edges are often not directed further increasing the difficulty in obtaining accurate graph topologies. In short, performing analysis regarding online social networks is difficult due to the size of the network together with acquiring accurate information regarding the formation of the network.

Gjoka *et al* (2010) make use of random walking to sample the characteristics of a number of social networks. The authors highlight that the use of random walking is not without its drawback. The graph structure can create distortions in the estimates by

“trapping” the random walk inside a local sub graph. Where instances of the random walk becoming trapped are seen the estimates of graph characteristics may be seen as inaccurate if the structure of the sub graph in which the walker is trapped does not share the same characteristics as the entire graph. The work makes use of a number of random walkers to mitigate the risk that any single walker may become trapped in a local sub graph. In addition the work considers a “frontier sampling” technique. The frontier sampling methodology also makes use of a number of random walks with the distinction (to the multiple random walk strategy) that the frontier sampling methodology selects start points for the random walks uniformly from a selection rather than the simple selection of a vertex in the graph at random. The work highlights that frontier sampling provides more robust estimations than simple random walks.

A high proportion of research activity concerning social networks has to date concerned either the reach-ability of the network or the generation of other metrics to gauge the connectivity of the network. Whilst complete network information is not available subsets of social networks provide for an accurate model of real world behaviours. Shi *et al* (2008 p.61) suggests, “For a variety of online networks, small subsets of vertices are relevant for efficient algorithms and dominate various graph and statistical properties. Frequently, these smaller subsets or *graph synopses* are easier to study and to understand”. De Meo *et al* (2012) study the centrality of social networks highlighting the use of ‘betweenness centrality’ in existing literature. Betweenness centrality relies on the notion that in social networks information flows along shortest paths. Therefore a node/edge has a high betweenness centrality if a large number of shortest paths cross it. In many ways the notion of betweenness centrality shares properties with that of the transit node (Bast *et al*, 2006) principle identified in the previous section where shortest path connecting distance locations pass through a limited subset of network vertices. Alahakoon *et al* (2011) highlight that existing methodologies for calculating betweenness centrality remain unsuitable for large graphs consisting of millions of vertices and edges. The work of De Meo *et al* (2012) makes use of random walking to calculate the betweenness centrality of edges in a social network. The authors propose using simple self-avoiding random walks of a maximum

length. The use of a maximum length for the walk is used to replicate the notion that the greater the ‘distance’ separating two vertices the less likely those vertices are to influence each other. Equation 2.7 formulates the betweenness centrality metric on a graph $G=(|V|,|E|)$ for node v . De Meo *et al* (2012) report that the approach they introduce is able to calculate the centrality metrics in $O(k|E(G)|)$ time, where k is the maximum length of the random walk and $|E(G)|$ is the number of edges in a graph $G=(|V|,|E|)$.

$$C_{b_n}(v) = \sum_{s \neq v \neq t \in V} \frac{p_{(s,t)}(v)}{p_{(s,t)}}$$

Equation 2.7 De Meo Centrality Metric

Where $p_{(s,t)}$ is the number of shortest paths connecting s to t

And $p_{(s,t)}(v)$ is the number of shortest paths connecting s to t passing through v

s = The path source; t = The path target

The New York Times (2012) highlights that low latency requirements of online social network such as quickly identifying links between individuals etc; a task that involves shortest path analysis. Gubichev *et al* (2010) perform a analysis of shortest path methods on a number of social networks. The methodology employed in the work shares similarities with that of the ALT* method of Goldberg and Harrison (2005) introduced in a previous section. In a variation to the ALT* approach however the authors, during the pre-computation stage, store details of the entire path generated between a vertex and the selected landmark vertices. Gubichev *et al* suggest that the diameter of social networks is likely to be small and therefore the storing of such paths is feasible. Other variations are introduced to the basic approach of the ALT* algorithm. The authors highlight that the social networks are more likely to see cycles and therefore introduce a mechanism to remove them as and when they occur. In a second modification the authors introduce shortcutting mechanism an approach seen in both the

highway and contraction hierarchy approaches. Consider the basic approach of the ALT* algorithm. Several landmark vertices are selected and the shortest path between each node and every landmark generated. The shortcutting mechanism highlighted by Gubichev *et al* is to perform a simple check to determine if a direct link can be found between a vertex on the approximate shortest path and all vertices occurring later on the path. Where possible such shortcuts are introduced reducing the length of the path. The approach is tested against a series of graph representing on-line social network gathered using crawling techniques. The mechanism developed. As with the method of Goldberg and Harrelson will often return sub optimal results. That is to say, the paths produced will be considered approximations of the shortest paths.

Tretyakov *et al* (2011) highlight that the appeal in the use of landmark based algorithms lies the speed of processing and general scalability offered by the approach stating “they have been shown to perform well in practice, scaling up to graphs with millions or even billions of edges with acceptable accuracy and response times of under one second per query”. The authors cite the work of Das Sharma *et al* (2010 p.406) who perform a web crawl of the Yahoo™ web network in order to generate a graph with the dimensions of 65,581,675 X 2,371,215,893 where the vertices represent distinct URLs gathered during the web crawl and the edges represent the number of links connecting those vertices (or web pages). The authors (Das Sharma *et al*, 2010) aim to identify shortest paths in network in terms of the diameter of the path as opposed to a provided cost metric as identified in the previous section. Also highlighted is that the size of the graph limits the feasibility of on-line processing. Das Sharma *et al* do not provide detailed analysis of the performance of their algorithm in terms of run time instead choosing to concentrating on the quality of the approximation that their algorithm produces, they do however state “a disk seek takes several milliseconds while the subsequent processing of the sketches takes only microseconds”. The size of the graph used requires the various lookup tables they use representing the lookup table between landmarks and other vertices to be stored on disk. Tretyakov *et al* (2011) make use of shortest path trees to reduce the cost of generating the shortest path after the calculation of the shortest distance. In effect, the work aims to remove the computational overhead

that is prevalent in advanced shortest mechanism described in the previous section as the “unpacking” mechanism. They report being able to perform shortest path queries on the Skype graph highlighted in Table 2.5 in an average of 16.25 milliseconds. The algorithm described makes use of shortcutting mechanisms similar to those of Gubichev *et al* (2010)

Agarwal *et al* (2012) propose the use of “vicinity mechanisms” to identify shorter paths on large scale graphs representing social networks where the vicinity of a vertex is a carefully selected subset of its neighbours. During an offline phase, for each vertex u in the network information regarding a certain subset of vertices in the neighborhood of u is computed and stored. During a subsequent online phase those vicinities are used to compute the shortest paths using the idea of vicinity intersection. The authors give the exacting requirements for their definition of a vicinity, detailing three core requirements. First, vicinities must guarantee correctness. If vicinities of s and t intersect one of the nodes that lies in this intersection must be on the shortest path between s and t . Second, vicinities should be large enough so that most of the source-destination pairs have intersecting vicinities. Finally, vicinities should be small enough so that the memory requirements are reasonable. The identification of vicinities is far more complex than the identification of nearest neighbour nodes. The approach makes use of a modified all pairs shortest path algorithm identified in Thorup and Zwick (2005). Other than the PHAST approach detailed earlier all other approaches review attempt the point to point rather than the all pairs problem. The method of Throup and Zwick is similar to that of Abraham *et al* (2011) which the network split into a series of cell with each containing a maximum number of vertices. Where a vertex is common to the vicinity of two vertices the shortest path can be identified using lookup tables. In order to generate the vicinity a set of nodes (denoted by L) is generated by sampling each node in the network with a probability proportional to its degree. The vicinity of each node u to be the set of nodes v , such that distance between u and v is no more than the distance from u to its closest node in L . Equation 2.8 is used to generate the probability of a node being included in set L for a graph $G = (|V|, |E|)$. The authors report results of 0.3 milliseconds on a Flickr based graph of 1.72 million vertices X

22.61 million edges. However, it is interesting to note that no note is made of what happens when neither intersection process fails, i.e. there is no intersection between two vertices therefore making the identification of the shortest path impossible.

$$P_s = \frac{|E|}{\partial|V|\sqrt{|V|}} \cdot \left\lceil \frac{2|E|}{|V|} \cdot d(u) \right\rceil$$

Equation 2.8 Agarwal Probability Metric

Where $|E|$ is the number of edges in the graph G

And $|V|$ is the number of vertices in the graph G

And ∂ is a user defined parameter

And $d(u)$ is the degree of vertex u

This section has highlighted the scale of social networks. It has primarily been concerned with on-line social networks however it should be noted that similar challenges are faced when handling off line instances of social networks such as in the form of citation datasets such as the DBLP database (Tretyakov *et al*, 2011). In many cases the analysis of social networks involves not only the calculation of the shortest paths between two vertices but also the generation of various graph centric metrics such as centrality. A major difficulty in performing analysis on social networks can be seen in the difficulties faced in obtaining the basic graph structures. Where obtained the networks are an order of magnitude larger than other real world instances such as road networks and in addition exhibit a higher density. The size and density properties of social network require alternative methods of analysis when compared to transportation network. Many algorithms (Das Sharma *et al*, 2010; Tretyakov *et al*, 2011; Gubichev *et al*, 2010) forego the calculation of exact shortest path and instead attempt to generate an approximation of the shortest path overcoming the issues that the increased size and density of the networks present.

2.7. Multi Criteria Path Optimisation

The introduction to this thesis highlighted how the real world approach to path finding is multi criteria in nature. In reality individuals will forego travelling along a shortest or quickest route for any number of reasons be it knowledge of road conditions, simplicity of alternatives or reasons personal to the decision maker. In the following section an attempt is made to formalise the multi criteria path optimisation problem. The basic concepts of Pareto optimality are introduced and historical approaches to the problem identified from the literature highlighted and reviewed. Where more than a single criterion is under consideration rather than a single result or ordered set the result will be a set consisting of solutions where it can be said no other solutions in the search space can be considered as being superior to them “when all objectives are considered” Zitzler (1999 p. 5). Such solutions are considered “Pareto Optimal”. A multi-criteria problem consists of a set of criteria or parameters that have to be maximized (or minimized) – that is a series of objectives that require optimisation.

The concept of Pareto dominance may be considered as being central to effective performance of multi-objective optimisation given the often conflicting nature of the problems. In such a case there is no single solution point that yields the "best" value for all objectives. Instead the best solutions, a Pareto optimal or non-dominated set, are a group of solutions such that selecting any one of them in place of another will always sacrifice quality for at least one objective while improving at least one other. The concept of Pareto optimality is illustrated in Figure 2.27. Pareto optimality may be described as *'the best that could be achieved without disadvantaging at least one group'*. Any two of the solutions in a search space can be taken from the problem search and each checked for dominance against the other. For any pair of solutions it should be easy to detect which is the better solution or if the solutions are mutually indifferent. In terms of Pareto optimality accepting a dominating solution will mean admitting an

inferior solution and as a result degrade the solution space. The key to this method of optimisation is that Pareto based non-dominated solutions are welcome whilst dominated solutions are unwelcome. Using Algorithm 2.8 the Pareto optimal front can be extracted from the search space represented by a series of solutions.

Steuer (1986) suggests that for a problem having more than a single optimisation objective solution x^1 is said to dominate the other solution x^2 if both of the following conditions are true:

- a) The solution x^1 is no worse than x^2 in all objectives
- b) The solution x^1 is strictly better than x^2 in at least one objective

Steuer (1986)

If either of the above conditions is not met then the solution x^1 does not dominate solution x^2 and can be considered as an example of a solution that may be of possible interest to the decision maker. In Figure 2.28 a single four-solution search space is presented. In the suggested example solution x^4 dominates solution x^1 , x^2 and x^3 as the solution x^4 is better in both of the criteria. Solution x^1 is also dominated by solutions x^2 and x^3 . Based on these solutions the feasibility set would consist of a single solution x^4 . In the absence of x^4 the feasibility set would consist of x^2 and x^3 . Algorithm 2.8 (Deb 1998) shows how the above conditions can be applied to a set of potential solutions in order to identify the Pareto optimal fronts. A given solution is said to strongly dominate over another when solution x^1 is *strictly* better than solution x^2 in each of the criteria being considered. Within a given set those populations that are not strongly dominated by another member of the population are said to be weakly

dominant. However the Pareto optimal front will always consist of those solutions that are non-dominated. On discovery of a dominated solution, that solution can safely be removed from the result set. Horn (1996) states that the Pareto set will be the optimal set of results – the best that can be achieved without biasing any objective. The contents of Figure 2.28 are provided in textual form in Table 2.6.

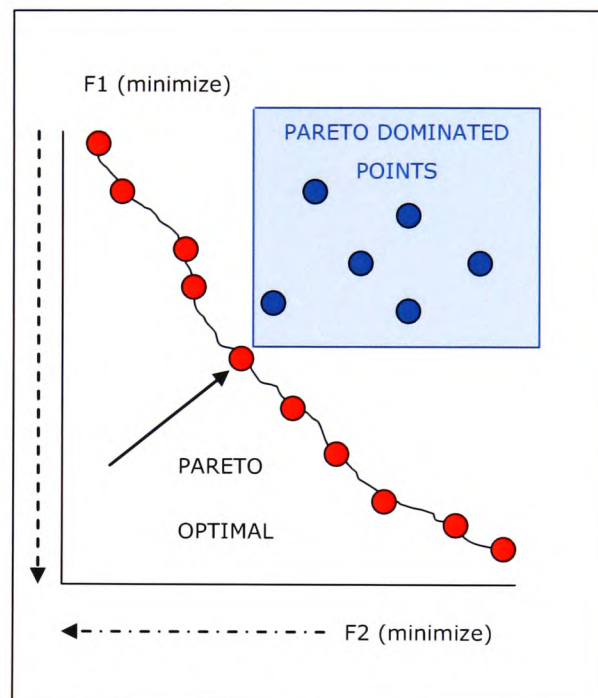


Figure 2.27 Example Pareto Optimal Front and Space

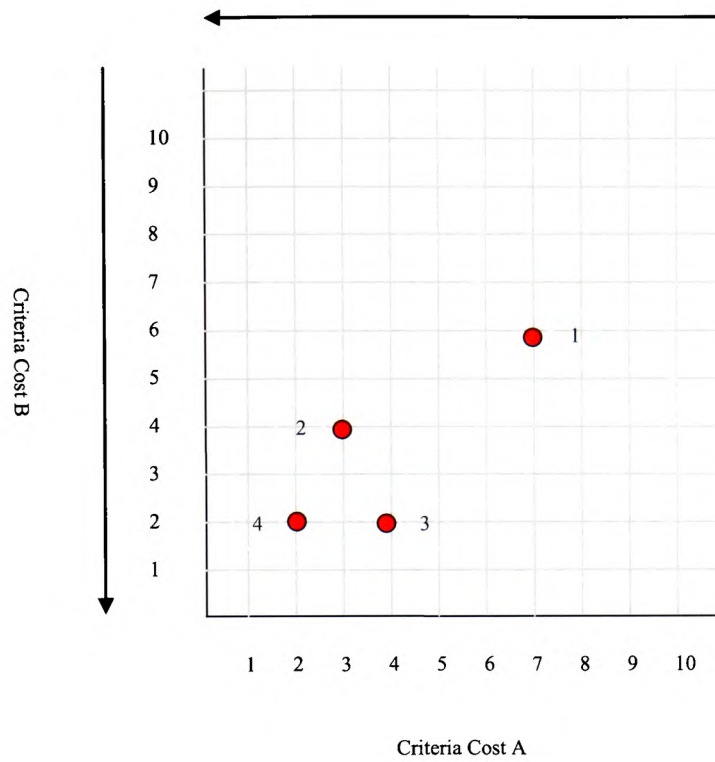


Figure 2.28 Example of 2D Search Space with Four Solutions

Solution	Criteria Cost A	Criteria Cost B
Solution 1 (x^1)	7	6
Solution 2 (x^2)	3	4
Solution 3 (x^3)	4	3
Solution 4 (x^4)	2	2

Table 2.6 Criteria Costs of Solutions Provided in Figure 2.28

Algorithm:	Pareto Front Extraction
Input:	S = Set Of Solutions To Be Checked For Dominance
Output:	O = Set Of Optimal Solutions
O = Empty List Of Optimal Solutions $COUNT = S $ FOR ($i = 0$ TO $COUNT$) { FOR ($j = 0$ TO $COUNT$) { IF ($i \neq j$) { IF ($S[i]$ Is Dominated By $S[j]$) Mark $S[i]$ As Dominated } } } } FOR ($k = 0$ TO $COUNT$) { IF ($S[k]$ Not Marked As Dominated) $O = O + S[k]$ } }	

Algorithm 2.8 Pareto Front Extraction Algorithm

In Horn (1996) the author presents the following typology for multi-criteria analysis:

Model 1: Decision before Search

Model 2: Search before Decision

Model 3: Iterative Search and Decision Process

The first of Horns' three models has already been considered as part of this work. In Chapter 1, the concepts of aggregation and weighting functions to reduce a multi criteria problem into a single criteria problem were introduced. The imperfections in doing so are clear. Corne *et al.* (2003) suggests that transforming a multi-criteria problem into a single-criterion problem is in many cases quite a radical simplification of the problem. Deb (2001) suggests that a Pareto optimal solution may be returned from

such a technique, but not the entire set of possible solutions. Horn (1996) suggests that while such techniques may be useful in satisfying a single decision maker, satisfying all is unlikely. The second and third of Horn’s models are closely aligned with the concepts of Pareto optimality. In these models the importance of any criteria will be unknown and instead a search is conducted for possible results before passing these to the decision maker (DM) for further analysis (search) or decision-making.

2.7.1. Worked Example of the MSPP

Having introduced the principles of Pareto optimality and dominance the following section applies those principles to the path planning problem. This is done initially through a worked example before the existing literature and methodologies for the solution are considered. Figure 2.29 introduces a simple graph that is used to demonstrate the principles of Pareto dominance and its application to the shortest path problem. The graph illustrates a directed graph with various bi-criteria edges. In the following example the edges are referred to as distance and time although these are human labels attached for clarity purposes only. No scale is inferred. Multiple edges between vertices are allowed. Assuming that the vertices *A* and *D* are the source and destination vertices respectively, numerous paths between those two vertices are present. Table 2.7 details each of those paths together with the cost along each of the criteria.

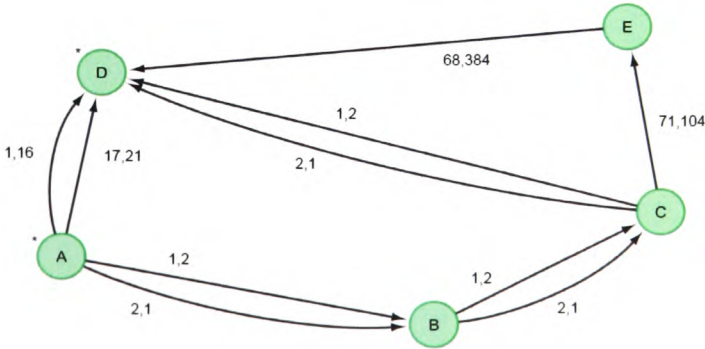


Figure 2.29 Example Two Criteria Graph

Path	Distance	Time
A_D	1	16
A_D	17	21
A_B_C_D	3	6
A_B_C_D	4	5
A_B_C_D	4	5
A_B_C_D	4	5
A_B_C_D	4	5
A_B_C_D	5	4
A_B_C_D	5	4
A_B_C_D	5	4
A_B_C_D	5	4
A_B_C_D	5	4
A_B_C_D	6	3
A_B_C_E_D	141	451
A_B_C_E_D	142	450
A_B_C_E_D	142	450
A_B_C_E_D	143	449
A_B_C_E_D	143	449

Table 2.7 All Paths Between Vertices A and D on Example Graph

Figure 2.30 demonstrates graphically what is made clear in Table 2.7 - Notably that there are two separate groupings - Those which passing through vertex *E* and those which do not. Those paths passing through vertex *E* are based on Steuers' (1986) definitions of Pareto optimality, dominated and therefore rejected. The removal of the dominating set of solutions results is highlighted in Table 2.8 and Figure 2.31. It is highlighted that several paths can be seen with the same traversal costs. Figure 2.31 presents the path values seen in the graph excluding those paths that pass through the node *E*. The table has also been updated to number the paths travelling *A_D*. The first of those paths has the cost set {1,16}. The second has the set {17,21}. Based on the previously given definitions of dominance, path #2 is clearly dominated by each of the other paths. The following Table 2.9 and Figure 2.33 indicate the Pareto optimal paths and front. Figure 2.33 represents the example graph with non-optimal paths removed. Note that as indicated in Table 2.7 multiple solutions may occupy the same space and therefore are obscured in Figure 2.32. Figure 2.32 and Figure 2.33 present the optimal

paths through the graph between the selected nodes. Any path chosen from Figure 2.33 will have a value pairing shown in Figure 2.32. Table 2.8 and Table 2.9 have been updated to remove duplicate path costs as seen in Table 2.7.

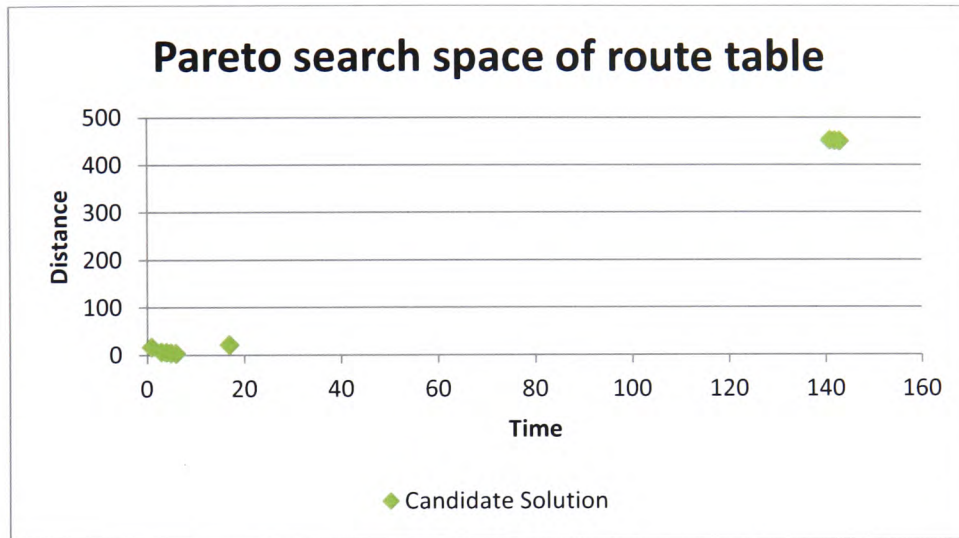


Figure 2.30 Pareto Search Space of Table 2.3

Path	Distance	Time
A_D#1	1	16
A_D#2	17	21
A_B_C_D	3	6
A_B_C_D	4	5
A_B_C_D	5	4
A_B_C_D	6	3

Table 2.8 Pareto Table with Paths Through Vertex E Removed

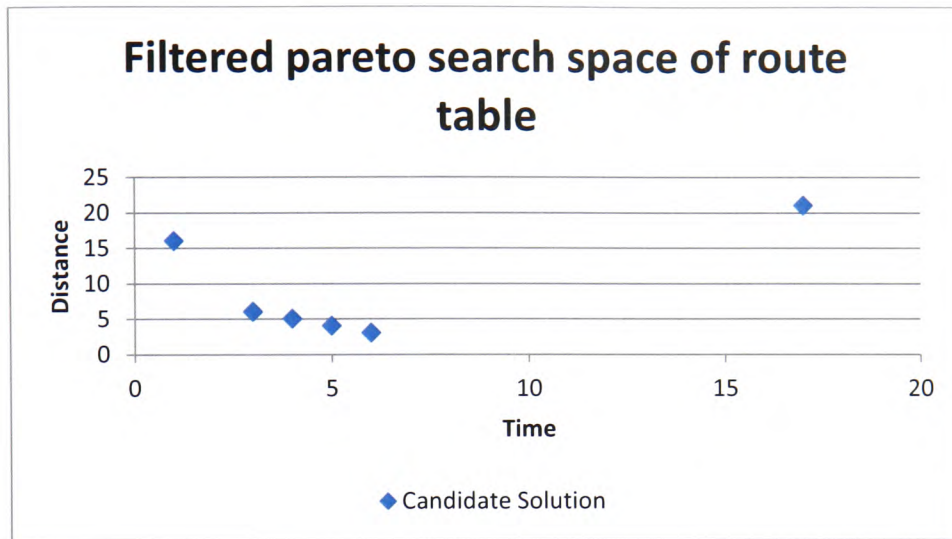


Figure 2.31 Filtered Pareto Search Space

Path	Distance	Time
A_D#1	1	16
A_B_C_D	3	6
A_B_C_D	4	5
A_B_C_D	5	4
A_B_C_D	6	3

Table 2.9 Pareto Optimal Paths

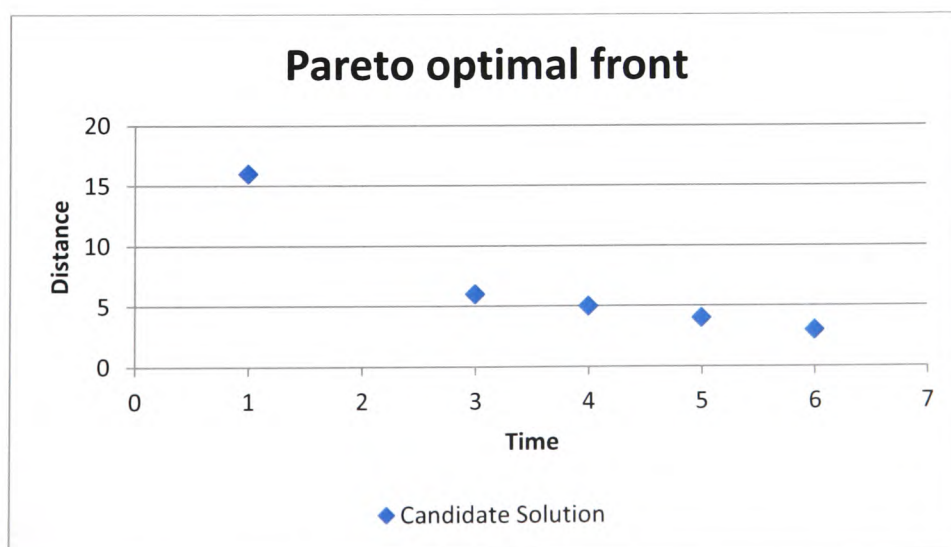


Figure 2.32 Pareto Optimal Solutions for Example Graph

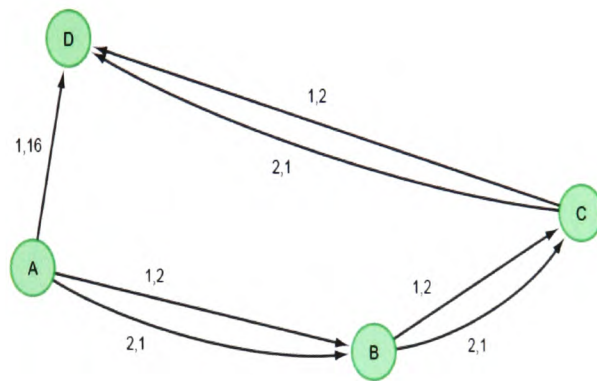


Figure 2.33 Pareto Optimal Paths Through Graph

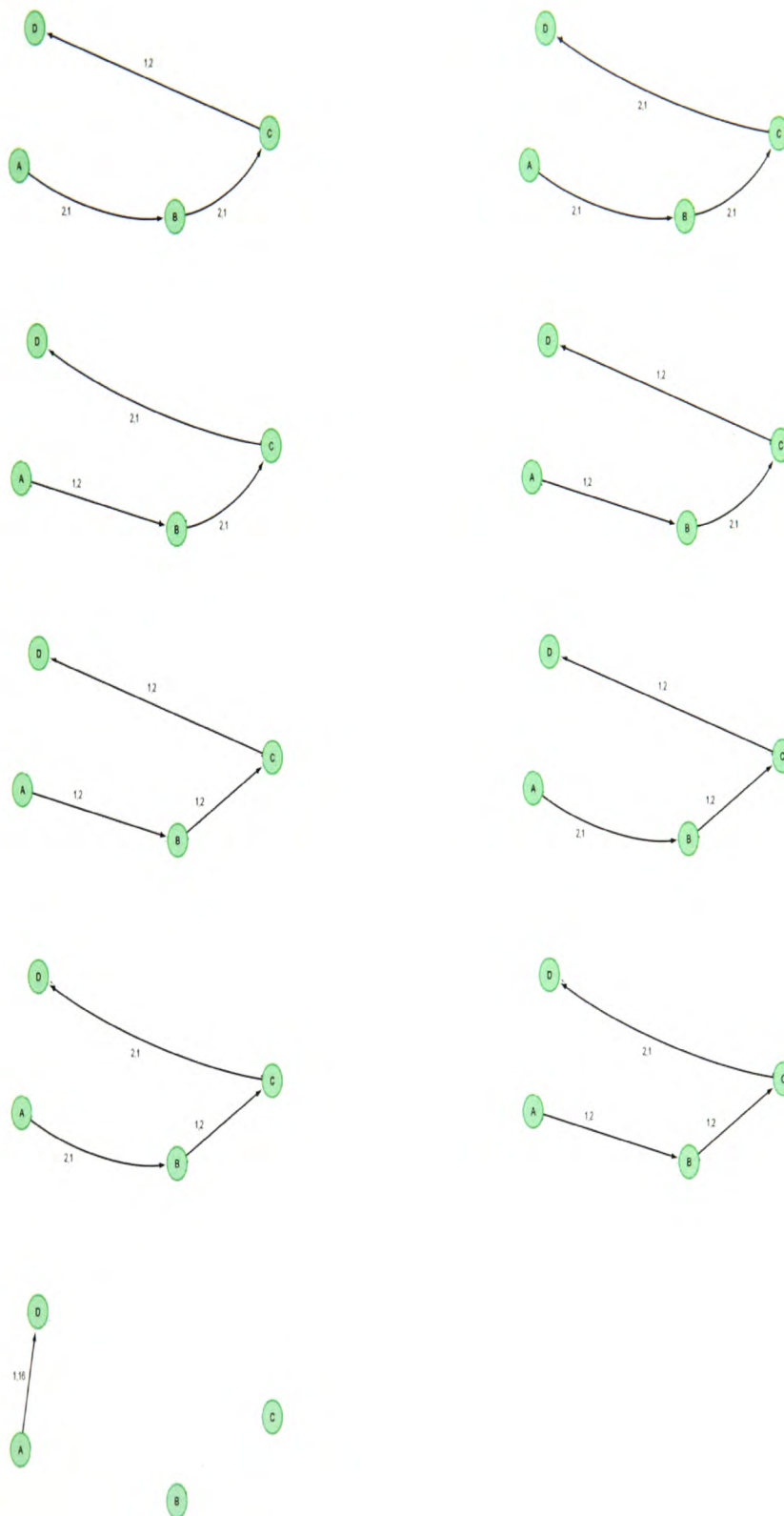


Figure 2.34 Individual Unique Paths Through Example Graph

The worked example provided concludes by highlighting the Pareto optimal path A_D travelling directly between the vertices. The cost set of the path gives a search space location of $\{1,16\}$. It should be noted that it is cheaper than the other possible solutions in one of the criteria (distance) but more expensive in the other (time). The path is not dominated by the other solutions. The solution should be presented to the decision maker (DM) for the feasibility of that solution to be considered. Figure 2.34 presents each of the unique paths highlighted in Figure 2.33. There are nine optimal paths present eight of which travel through the set $A_B_C_D$.

2.7.2. Existing Work on the MSPP

Having presented the concepts of Pareto optimality and dominance, and having related the same to the path planning process attention now turns to existing work in the field of the multi objective shortest path problem (MSPP). The multi-criteria nature of the path planning process has been made clear elsewhere in this work most notably the introduction where an in-depth discussion in to the nature of decision-making in path planning was undertaken. Having seen that the process of path planning is one that is multi criteria in nature it is perhaps interesting to note the majority of research into the path optimization problem has considered only a single objective. The reasons for this are entirely logical. The process of path optimization where more than a single objective is to be considered is an example of a problem that is NP hard (Garey and Johnson, 1979 and Hansen, 1979). Granat and Guerriero (2003) provide a detailed rationale for the NP-completeness of the issue.

Figure 2.17 highlighted that the amount of research undertaken into single criteria analysis has continued to grow since the turn of the century. Producing a similar diagram for the multi objective path optimization process would prove to be difficult. Firstly, whilst it may be possible to view the terms objective and criteria as

having the same meaning the lack of standardization within the literature results in search queries returning differing and often-disparate results. Secondly, the term “shortest path” may have little meaning when considering path optimization where more than a single criterion is to be optimised given the nature of the problem. Many authors whilst looking for shorter paths do not consider the problem as being directly an example of shortest path problem and so terms such as “shortest path” are absent from any key word sections related to that piece of literature. Figure 2.35 attempts to present the number of publications seen in the ScienceDirect online repository regarding the multi objective path optimisation process. The following keyword terms were entered into the ScienceDirect search mechanism:

- “Multi Criteria” AND Path
- “Multi-Criteria” AND Path
- “Multi-Objective” AND Path
- “Multi Objective” AND Path
- “Pareto” AND “Path”

The searches were performed both inside and outside quotations. The results of the above queries were then reviewed to extract those works where the either the shortest path or similar related problem is considered. The nature of what are considered similar problems is largely open to interpretation. For the purposes of this work similar works are considered any algorithms that attempt to perform routing or path finding are considered. Applications such as timetabling etc are not considered as routing applications for the purposes of this analysis, vehicle routing is however considered. The filtering step results in Figure 2.35 which highlights the number of publications related to multi objective path optimisation problems seen since the year 2000. Of the 863 publications seen to match the search terms “*Multi Objective*” AND *Path* published in 2012 only 26 may be considered as related to the multi objective path planning

process. In total across all key words terms 51 publications from 2012 are seen to be as *related* to the multi objective path planning process in its various forms. As has previously been suggested the exact nature of what is considered a related publication may be open to interpretation. However, Figure 2.35 highlights that when compared to Figure 2.17 considerably less research effort has been undertaken into the path planning when handling multiple objectives. Since the year 2000 approximately 325 research papers have been published on multi objective path optimization in comparison to 1,618 where only a single criterion is considered. Of interest is the general growth in the level of interest seen in the multi objective path-planning problem with 15% of the publications since 2000 becoming available in 2012 compared with slightly less than 3% in 2000. It should be noted at not all publications will consider the specific application of the MSPP.

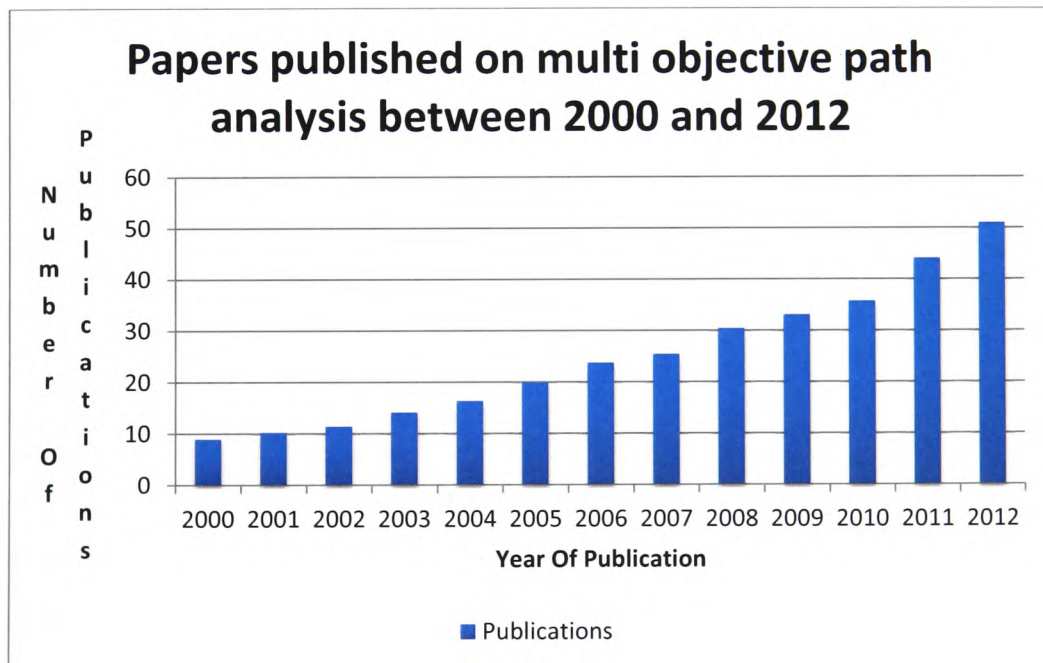


Figure 2.35 Publication on Multi Objective Path Analysis Since 2000

The work of Hansen (1979) was the first to formalize the problem of the MSPP and is referenced frequently in the literature. In that work he provides a label-setting algorithm for the MSPP problem. From the literature it is possible to identify three separate methodologies for the solution of multi objective shortest path problems following a typology introduced in Cohon (1978) each of these strategies will vary in how they search the problem space and are briefly discussed below.

Generating Methods

Methods utilizing generating methods will attempt to discover the complete Pareto optimal set or make a close approximation to it. When applied to the MSPP Hallam *et al* (2001, p.134) state that the generating method may have limitations. They argue “However, they do not work well for large graphs as there may be a very large number of such paths, and the necessary computations are beyond the acceptable scope of most computers.”, they present Climaco and Martins (1982), Hansen (1979) and Martins (1984) as examples of such methods. Bandyopadhyay (2008, p.270) suggests: “Note that theoretically, the number of PO solutions can be infinite. Since the ultimate purpose of an MOO algorithm is to provide the user with a set of solutions to choose from, it is necessary to limit the size of this set for it to be usable by the user”.

Methods Based on Utility Functions

Utility functions aim to introduce some notion of the decision makers (DM) preferences into the MSPP. Granat and Guerriero (2003 p.104) suggest that the use of utility functions reduces the problem into a single criteria methodology stating, “It is worth noting that in these methods the multi-criteria optimisation becomes a single-objective optimisation problem”.

Interactive Methods

Interactive Methods involve direct interaction with the decision maker. A set of solutions is offered to the DM, and through interaction their preferences are selected which leads to an iterative style process. Current *et al* (1990) suggest the goal of this approach is to assist the decision maker in selecting the preferred or best compromise solution from among the non-inferior solutions; they further highlight the appeal offered by interactive methods given the computational complexity of the problem. As only a subset of the optimal solutions are sought less processing time is required.

Based on the three solution methods previously given, users are presented with either a partial or complete set of solutions representing the Pareto optimal front. In the case of the latter, it is suggested the methodology appeals because:

*“Although the calculation of the whole set of the non-dominated solutions in the bi-objective case can be done easily, it must be remarked that the number of the non-dominated solutions can be very large. So, this is not, in general, an effective way of presenting alternative choices to a decision maker”¹ (Coutinho-Rodrigues *et al*, 1999, p.790).*

Ziontis and Wallenius (1976, p.662) state that "managers seem to find it easier to respond to the trade-off questions in the context of a concrete situation (tradeoffs that are attainable from realizable situations) rather than in an abstract situation." i.e. they (managers) find it easier to select from options rather than defining parameters. Granat & Guerriero (2003 p.104) propose the generating and utility function methods are insufficient, for a number of reasons stating:

“First of all, generating the whole Pareto-optimal solution set may be computationally intractable, even in the case when a small number of criteria is considered. Furthermore, supposing the Pareto optimal solution set has been

determined, it may not be easy for the decision maker to choose from a very large set the non-dominated path that he/she likes best, since the number of Pareto-optimal solutions may grow exponentially with the number of nodes. Secondly, it is difficult for the decision maker to define a utility function representing his/ her preferences”.

Skriver (2000) presents the following (Figure 2.36) taxonomy of the various approaches to the MSPP that can be seen in the literature.

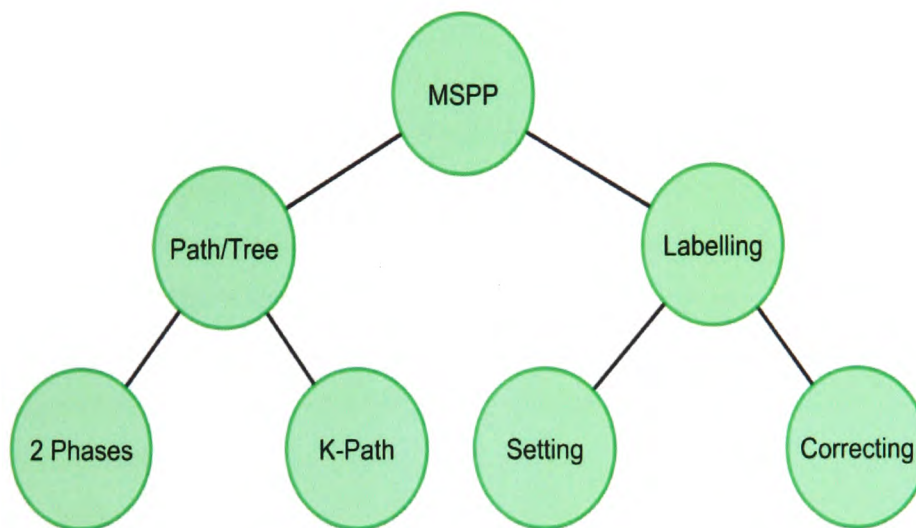


Figure 2.36 The Skriver Taxonomy of MSPP Techniques

Bezerra (2013, p.346) highlights that a number of exact algorithms can be identified for handling bi-criterion shortest path problems. The authors however proceed to highlight that attempts to move to numbers of criteria greater than two are limited stating, “this generalisation is not trivial”. Bezerra *et al* make use of ant colony optimization for the solution of MSPP. The previously cited work of Hansen (1979) provides an example of the label setting approach to the solution of the MSPP. The Hansen (1979) methodology is extended in the works of Martins (1984) and Tung and Chew (1982). Skriver (2000) and Skriver and Anderson (2000) present an example of the label correcting approach. That work is heavily influenced by the work of Brumbaugh-Smith and Shier (1989). Brumbaugh-Smith and Shier introduce an

algorithm in which multiple Pareto set merge operations are conducted as the graph is explored. Skriver and Anderson (2000) extend the algorithm highlighting the performance restriction that the multiple Pareto merge operation causes. The authors propose that inducing some simple domination conditions on the edge-candidates should make it possible to discard ‘expensive’ edges as soon as possible. Two methods to reach this goal are suggested. The first involves performing successive runs of Dijkstra’s algorithm on each of the two criteria being considered. They also suggest the implementation of an ‘over-take’ method which appears to combine basic Pareto optimality checks with the relaxation method of Dijkstra’s algorithm and is based upon the method highlighted by Tung and Chew (1982).

Climaco and Martins (1982) introduce the use of a combination of the Dijkstra’s algorithm and the K Shortest path approach. The Dijkstra algorithm is applied to one of the two criteria under consideration (path cost D). An implementation of Lawler’s K shortest path algorithm is then applied to the second criteria and paths of increasing value generated until the cost of the first criteria on Path K exceeds that of path cost D . Climaco and Martins (1982) propose that the methodology will return the entire set of Pareto optimal paths between two nodes. Skriver and Anderson (2000) contend that the K-shortest path (and indeed any path/tree method) approach will always be out performed by algorithms using the labelling approach. In Rhaith and Erghott (2009), however this is demonstrated to be partially untrue. They demonstrate that in many cases the labelling method is out performed by an implementation of a near shortest path algorithm first suggested in Carysle and Wood (2005) in many cases of real world graph instances. It should be noted however that the test conducted by Rhaith and Erghott (2009) reported some ‘failures’, demonstrated by exceptionally long run times. The reverse is true on random graphs where the labelling method is always the optimal algorithm as proposed by Skriver and Anderson (2000).

A K shortest path algorithm is also utilized in the work of Coutinho-Rodrigues *et al.* (1999). The authors extend the previous work of Coutinho-Rodrigues and Climaco

(1994) in which a logistics decision support system is introduced. The later work builds upon that decision support system. The system initially determines the first three non-dominated solutions which are presented to the user. The user is then given the option to search within any two of those points. Figure 2.37 presents the result of the initial search to the user. Having selected points 1 and 2 to search between, the users would then be presented with the results highlighted in Figure 2.38. Coutinho-Rodrigues *et al.* (1999 p.793) use a constrained K shortest path algorithm to identify “automatic calculation of the whole set of non-dominated solutions inside the gap”. The constraints are formed from the minimum and maximum path costs required in each criteria. The interactive nature of the process also allows the search to terminate on user interaction. Granat and Guerriero (2003) make use of a ‘reference point’ interactivity model which has the same aspirations. In both the work of Coutinho-Rodrigues *et al.* (1999) and Granat and Guerriero (2003) random graphs are used in the test case. In the latter work, the graphs are large and very well connected i.e. between 5,000 and 40,000 nodes and 1,000,000 edges. However, it is also of interest with regard to the number of criteria where four and eight criteria sets are used. In the majority of cases covered the criteria sets are limited to bi-criteria problems or in limited cases tri-criteria (Pinta and Pascoal, 2010).

Modesti and Sciomachen (1997 p. 495) make use of an interactive method to solve the MSPP when applied to multi-modal transport plans. The authors require users to specify preferences towards differing modes of transport that are combined with Dijkstra’s shortest path algorithm in order to present a subset of the optimal front. The authors state that they make use of a utility measure “taking into a proper account the different users’ propensities” in “order to overcome the multiplicity of Pareto optimal solutions”.

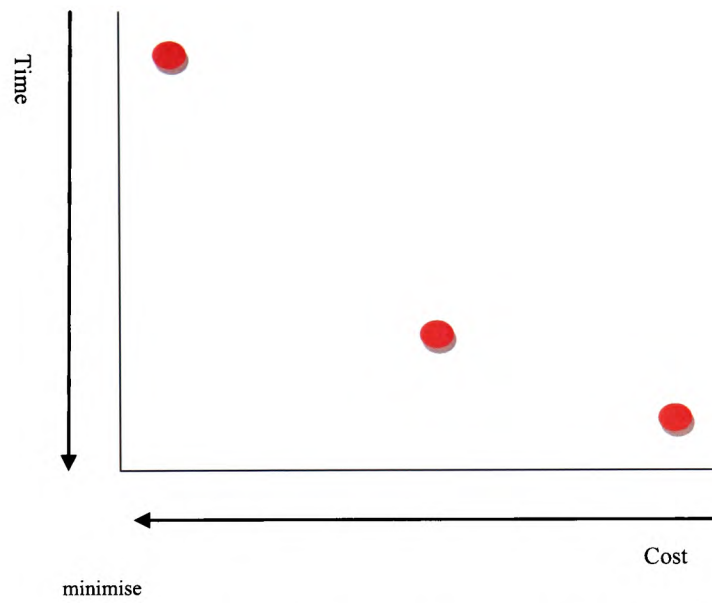


Figure 2.37 Initial Search Results (Coutinho-Rodrigues et al. (1994))

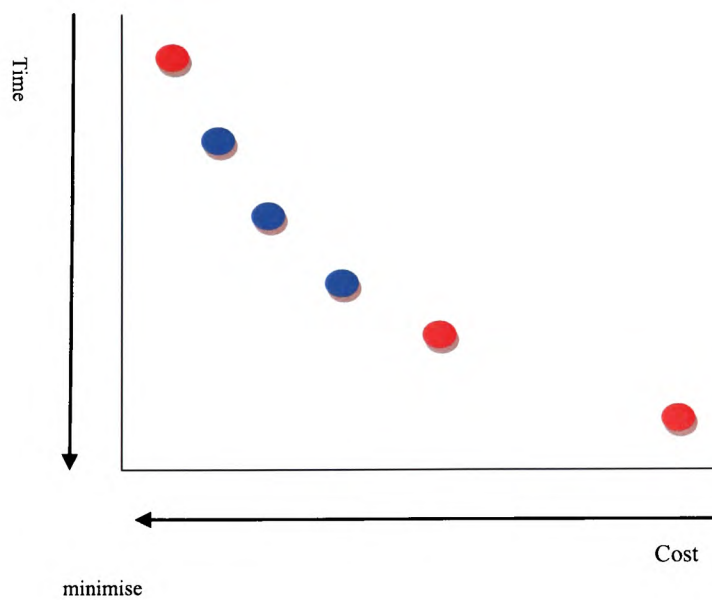


Figure 2.38 Refined Search Results (Coutinho-Rodrigues et al, 1994)

Muller (2010) presents a constrained heuristic function for solving the problem of the vehicle routing problem. In that work the author makes use of an ‘e-constraint’ where one of the criteria is selected as the primary criteria with additional criteria then act as constraints. As an example considering a bi-objective problem, Criteria A would be subjected to optimisation; criteria B would then form the basis of a constraint model. In the cited work an example of a vehicle routing problem is presented where the shortest distance travelled is required and certain delivery time constraints have to be met. Grabener *et al* (2010) extend the algorithm of Martins and Santos (1999) in order to present a time constrained multi modal transport planning tool. Delling and Wagner (2009) employ a variation of the Dijkstra shortest path algorithm. The authors report very promising results on large-scale real road networks, they also highlight that a certain degree of pre-processing of the datasets is required, alongside higher memory requirements. In addition the authors highlight that their attempts at generating the complete Pareto set on a graph of Western Europe failed because “it turns out this input is too big for finding all Pareto routes”. Instead they reduce the size of the graph and attempt to limit the size of the Pareto set. Delling and Wagner extend Mohring *et al* (2007) to present improvements to the single criterion Dijkstra algorithm through graph partitioning. The algorithms effectively limit the amount of the graph being explored. Sauvanet and Néron (2010 p.616) introduce what is suggested to be a novel method for reducing the number of solutions offered to a user. In their work a variation of the A* shortest path algorithm is developed which presents users with a series of non-dominated cycling paths, with the criteria considered being the distance and the ‘insecurity’ of the road link. The results of those experiments have been made available online (<http://www.geovelo.fr/>). The authors aim to produce a subset of optimal solutions stating “however computing all solutions can be time consuming and not necessarily helpful if the need is to obtain a compromise solution”. The authors reduce the problem down to a single criterion function using preference weightings. The user selects a preference using a between shortest distance or increased safety and their preference then translated into single value to which the A* algorithm is applied.

The work of Mandow and Perez de la Cruz (2010) presents the New Multi Objective A* (NAMOA*) algorithm, an extension of the work of Stewart and White (1991). In both of these works an implementation of the A* algorithm is presented in an extended form to solve the MSPP. Unlike the approach of Skriver and Anderson (2000) which may be described as a destructive technique (in that edges are removed from the graph until only the Pareto optimal paths remain) the NAMOA* approach maintains a separate graph structure, to which optimal paths are added. Manchua and Mandow (2012) present a study of the NAMOA* algorithm on a series of real world graphs made available as part of the ninth DIMACS (Discrete Mathematics and Theoretical Computer Science) challenge where time and distance may be combined to present bi-objective graphs. The graphs used are large scale, often of entire US states. The graphs used in that study are highlighted below in Table 2.10. The authors report varying results ranging from fractions of a second (0.12) to over 46 minutes on the same graph. Manchua and Mandow also apply a time limit of one hour and report a number of scenarios where that time limit is not met. The longest processing effort took over 40 hours leading to the introduction of the previously stated time limit. In Manchua *et al* (2012) the authors perform a comparison of various heuristic algorithms and heuristic values for the MSPP reporting that generally the NAMOA* is the most promising with certain heuristic functions.

Graph	Vertices	Edges
New York City	264,346	730,100
San Francisco Bay	321,270	794,830
Colorado	435,666	1,042,400
Florida	1,070,376	2,712,798

Table 2.10 NAMOA* Real World Test Graphs

Saadatseresht *et al* (2009) combined single criterion shortest path algorithms together with empirical evidence regarding building hazard metrics to produce models of evacuation zones. Coutinho-Rodrigues *et al* (2012) extend such an application to the wider scale problem of urban evacuation zones where distance to safety and minimization of perceived risk are optimised. The distance to safety was considered as a single criterion network analysis problem. Where Saadatseresht *et al* cover a small group of buildings as an evacuation zone Coutinho-Rodrigues *et al* cover town and city level evacuation plans. Dell’Olmo *et al* (2005) integrate a multi objective shortest path algorithm (MSPA) into a GIS in order to identify dissimilar yet optimal paths. The MSPA is based upon that of Martins and Santos (1999). Having identified the optimal paths the proposed methodology prunes those paths into a disparate set.

2.7.2.1 Heuristic and Evolutionary Approaches to the MSPP

Having considered the traditional algorithmic methods of searching for a series of Pareto optimal ‘shortest’ paths this work now turns to evolutionary and heuristic approaches. Such approaches are reasonably recent in the literature and as such publications are sparse. The algorithmic methods described up to this point have evolved over the course of twenty-eight years and still comparatively rare. The same can be said of heuristic approaches with an even greater degree of certainty.

Mnuemoto *et al* (1998) and Ahn and Ramakrisma (2002) each make use of genetic and evolutionary strategies for the solution of the single criterion path problem. Mooney (2004) and Mooney and Winstanly (2006) however extend those underlying techniques for the multi objective shortest problem, as do Gen and Lin (2004) where an evolutionary algorithm is used in combination with a fuzzy logic based system. He *et al* (2007) make use of similar mechanisms to those employed by Mooney and Winstanley (2006) to solve the MSPP using genetic algorithms. Unlike the approach of Mooney and Winstanley where random walking is used to generate the populations He *et al* use depth first search based mechanisms. The algorithm also makes use of Pareto ranking

and nicheing. Crichigno and Baran (2004) and Pangallinan and Janssens (2007) make use of a graph based implementation of the Strength Pareto Evolutionary Algorithm (SPEA) approach (discussed later in thesis) to solve the MSPP problem. Chakraborty *et al* (2005 p. 190) presents a Genetic Algorithm based approach to the car navigation problem, which attempts to produce “several alternate routes depending on different criterion according to driver's choice such as shortest path by distance, path which contains minimum number of turns, path passing through mountains or by the side of a river etc”. Liu *et al* (2012a) presents an Genetic Algorithm approach, which attempts to make use of spanning trees to increase the Genetic Algorithm effectiveness. Amongst the works making use of Genetic Algorithm based approaches the concept of elitism appears to be universal. That is to say each algorithm implements a secondary population consisting of optimal solutions. In addition binary method of selection dominate the literature as a means of selecting candidate solution for cross over and mutation (introduced formally in Chapter 3).

Liu *et al* (2012b) present a development of Liu *et al* (2012a). In those works the authors make use of simulated annealing and genetic algorithms respectively to solve multi objective shortest path problems. The works are of particular interest for two reasons. Firstly, the use of simulated annealing is rare in any application of multi objective analysis where a reduction to single criteria analysis via aggregation is not performed. Secondly, other than this thesis, the work of Liu *et al* (2012b) is a rare example where simulated annealing has been used to solve multi objective path optimization problems and perhaps importantly a comparison of genetic algorithm based approaches with other techniques using the same datasets is performed. Both works make of spanning tress to represent paths through the graph. Both algorithms are tested against a limited set of synthetic graphs. In addition the simulated annealing approach is tested against small example graphs representing a section of the road network in mainland China with the sizes 70 X 207 and 581 X 954. Tests on synthetic graphs are performed on datasets consisting of a high vertex to edge ratio. The smallest graph tested consists of 2000 vertices and 10,000 edges. The largest graph consists of 8,000 vertices and 76,069 edges. It should be noted that the ratio of the upper and lower

bounds of the connectivity of the graphs varies greatly with a minimum of three outgoing edges seen on the largest of the graphs, and a maximum of 35. The authors report limited success of the simulated annealing approach on larger sized graphs. However it should be noted that the measure of success used might be considered very limiting with only an exact replication of the optimal set of solutions being considered. They authors report that the simulated annealing algorithm was able to identify the complete set of optimal solutions in just 3.3% of cases. The implementation of a Genetic Algorithm approach was unable to identify a single complete optimal set. Previous work cited in this work have highlighted that a limited set of optimal will often be considered satisfactory, and therefore a looser definition of success may be considered as being more appropriate in comparison to that used by Liu et al. In addition the high ratio of vertices to edges used of the graph may be reduce the effectiveness of the simulated annealing technique. On smaller graphs (2,000 – 14,071) the Simulated Annealing approach slightly outperforms the Genetic Algorithm approach, with 53.33% of tests returning an exact match for the Simulated Annealing and 50% for the Genetic Algorithm. The high vertices to edge ratio seen in larger graphs is repeated in smaller graphs with a wide range of outgoing edges seen, with a minimum value of four and a maximum of 26. The nature of vertex to edge ratio seen in the synthetic test graphs is unlike that seen in real road networks in the form of roads as highlighted in Table 2.10.

Bezerra *et al* (2011) and Bezerra *et al* (2013) make use of ant colony optimization to assist in the solution of the MSPP. The authors were not the first to make use of ant colony optimization for routing applications with Baran and Schaerer (2003) making use of ant colony optimization (ACO) to assist in the solution of the vehicle routing problem with time windows. In their work of 2011 the authors (Bezerra *et al*) develop an ACO technique they name GRACE. Ant Colony Optimization is a bio-inspired meta-heuristic that uses the concept of swarm intelligence originally proposed by Dorigo (1992). Dorigo and Socha (2006) discuss a number of techniques that may be employed to extend the single criteria optimization technique to one that is capable of handling multiple criterions. The GRACE technique follows a two-phase approach. In

the first of the two stages the algorithm identifies the extreme ends of the optimal front using traditional shortest path analysis techniques. A process of iteration is then used to determine further possible solutions between the extreme solutions using ant colony simulation. The approach is tested against a implementation of multi objective genetic algorithm based upon the approach of Deb *et al* (2002) introduced in the following chapter together with an alternative ant colony optimization approach to the MSPP introduced in Hackel *et al* (2008). Both Bazerra *et al* (2011) and Hackel (2008) employ multiple ant colonies when considering additional objectives. Each colony employed by the system considers a single criterion. The authors' present results demonstrating better quality results from the ant colony approaches when compared to the genetic algorithm approach and show better results obtained using the GRACE approach than from that of Hackel *et al* (2008). The datasets used in the study are formed from either grid based graphs or complete graphs. In either of the two types the size of the graph is quite limited in comparison with other works such as that of Liu *et al* (2012a, 2012b) and range from 25-200 vertices for the complete graphs. In Bazerra *et al* (2013) the authors report that as the size of the graph increases then the underlying topology of the graph becomes less important to the comparative success of the methodology. The basic methodology algorithm used in Bazerra *et al* (2013) is that of the 2011 work by the same authors. A key difference in the 2013 work however is the analysis into the behaviours of the algorithms. Zhang *et al* (2012) make use of ant colony optimization technique for the solution of path planning problems for mobile robots in a discrete space.

Pahlavani *et al* (2012) make use of a technique they call 'invasive weed optimization' to assist in the solution of the MSPP. In the work the authors aim to solve the MSPP with two criteria where each criterion has been assigned a relative importance value. The technique of invasive weed optimization shares common goals with genetic algorithms formally introduced in the following chapter. The technique of Pahlavani *et al* (2012) generates a random set of solutions. The technique then calculates the fitness of each of solution. For each of the solutions in the population a number of 'seeds' is produced. The better the fitness of a solution the more seeds a given solution may produce. Each seed then gives rise to a related solution around the parent. Small

changes to the selected member of the population are introduced mimicking the process of weed dispersal. Each of the related solutions is then added to the end of the population. At the end of each iterations the population is sorted based upon the fitness of the solutions and the population truncated to its maximum length. Given the use of assigned weightings for a preferred method of travel method reduces the problem to a single criteria solution based upon an aggregation method. The authors test the approach against a multi phase approach to the Dijkstra shortest path algorithm where each phase represents a run against a single criterion, a genetic algorithm approach and the invasive weed approach. The authors find the Dijkstra approach completed the tests in around 18 seconds compared with 56 and 44 seconds for the Genetic Algorithm and invasive weed approach respectively. In addition to the use of the invasive weed approach the technique is novel due to the use of real world road networks as the basis for the test datasets. The road network is based upon that of Tehran, Iran and consists of 30,880 vertices and 34,951 edges. The authors of the work do not describe whether the graph was considered bi-directional. The values of the graph particularly the ratio between the number of edges and vertices (1.13) would indicate a non-directional approach was used which may affect the performance of the algorithms though the substantial reduction in the number of possible paths between two points. The nature of the runtime gives the appearance that the implementation of the Dijkstra shortest path algorithm was not optimised using any heap or bucket structure.

Horoba (2010) undertakes a review into the runtime of evolutionary algorithms for the shortest path problem. Horoba highlights that all previous studies into the runtime of evolutionary approaches to the shortest path problem have primarily concentrated on single criteria problems and hence Horoba is the first to consider a formulization for the runtime of evolutionary approaches where more than a single criteria is under consideration. The algorithm analysed by Horoba is based upon that of Doerr *et al* (2008) though expanded to handle multiple criteria. The Doerr *et al* algorithm initializes a population consisting of all paths where the length will be equal to zero. The mutation operator either adds or removes a vertex to the end of the path at random. In many ways the approach taken by Horoba (2010) produces an algorithm that operates more like a simplified version of the Pareto Archiving Evolutionary Strategy (PAES) developed by Knowles and Corne (1999) introduced in the following chapter

rather than a traditional population based approach to the evolutionary algorithm. The algorithm retains the population approach of the traditional evolutionary algorithm but performs the selection and reproduction operators on only a single member of that population. Unlike Doerr *et al* (2008) Horoba makes no use of a crossover methodology.

Davoodi *et al* (2013) together with Bezerra *et al* (2011) make use of the NSGA-II evolutionary approach of Deb *et al* (2002) for solving the MSPP. In the case of Davoodi *et al* (2013) the algorithm developed attempts to solve shortest path planning problem for mobile robots although the authors highlight the feasibility of the techniques to act as a artificial intelligence control in entertainment game software. The search space of Davoodi *et al* (2013) is represented as a regular grid, with obstacles to be avoided in the search space removed from the grid. The genetic chromosome is represented by a linked list giving the x and y coordinates in the regular grid. The Deb *et al* NSGA-II approach appears to be commonly used when considering evolutionary approaches to the MSPP, featuring also in the work of Hung *et al* (2007), which also addresses the application of mobile robot path planning using evolutionary approaches. Chitra and Subbaraj (2012) make use of the NSGA-II based technique for multi objective path planning in computer networks.

Fang *et al* (2011) make use of any colony optimization to solve the evacuation problem. The authors develop an algorithm to assist in the planning of evacuation routes from a stadium in the event of an emergency with evacuation routes being represented as directed links in a graph. The algorithm is tested against a K shortest path approach and an implementation of the NSGA-II of Deb (2002). The authors do not produce any quantitative metrics in terms of runtime for each of the algorithms and instead produce a qualities assessment of the results seen from each approach. The report notable difference between the results seen in each algorithm with the K shortest path and genetic (NSGA) approach resulting in higher numbers of people evacuated from the test

stadium but along more congested paths. The any colony based approaches produced a more even spread amongst the possible evacuation route.

2.8. Chapter Summary

The current chapter has discussed shortest and multi criteria path analysis. From the study of existing literature a series of significant facts may be drawn.

- Advances in single criterion path analysis continue to be made. These can be seen in methodologies such as graph partitioning and structural hierarchies. These have the effect of allowing continent wide shortest paths to be calculated extremely quickly (measured in nano seconds). The Santos (2006) algorithm allows large numbers of paths of increasing length to be generated in a matter of seconds on continent sized road networks.

- Existing literature has demonstrated the MSPP to have been the subject of sporadic research interest. Recent years have seen the renewal of interest in the subject area through the use of evolutionary algorithms. However, even where undertaken, research effort is often focused on the application of existing techniques to a given problem rather than the development of new solution mechanisms.

- Heuristic mechanisms for the solution of the MSPP have been limited to the use of evolutionary algorithms or techniques such as ant colony optimisation. Liu (2012 pp 3120) presents similar findings suggesting, “The SA has been widely applied in solving some route related problems such as VRP and TSP. To our knowledge, however, we do not find any approach for applying SA to MSPPs”. An example of alternative routing applications making use of Simulated Annealing using multi objective optimization can be seen in Banos *et al* (2012) who use a

simulated annealing technique for the vehicle routing problem, a variation of the travelling salesman application. Zidi *et al* (2011) present a variation to the same approach (Simulated Annealing for the vehicle routing problem).

- From an implementation perspective, algorithmic solutions to the MSPP appear relatively simple. Notable examples can be seen in the Skriver & Anderson (2000) method, and the approach taken by Climaco and Martins (1982). However in practice the solution of the MSPP is demonstrated to be intractable often requiring days (40 hours) to complete the generation of the optimal set where a single criterion problem may be solved in micro and nanoseconds.

Chapter Three: Multi-Objective Optimisation

3. Multi Objective Optimisation

The previous chapters have sought to formalize the problem under review. The current chapter will introduce the various algorithms that will form the basis for the experimental phase of this research. A comparison of Multi Objective Evolutionary Algorithms (MOEA) approaches is performed before moving on to consider the multi objective approaches to the Tabu Search and Simulated Annealing. The chapter ends with a brief review of various metrics used to review the quality of multi objective (MO) solutions.

3.1. Genetic Algorithms and Evolutionary Computation

Although conceived by Holland in 1975 it is the work of Goldberg (1989) that brought widespread awareness and acceptance of Genetic Algorithms to the fore. Goldberg (1989) highlights that Genetic Algorithms have been demonstrated to provide robust search in complex spaces and that Genetic Algorithms are increasingly finding their way into a wider range of applications. Holland (1975) presents Genetic Algorithms in terms of a simple binary string chromosome and population in the environment E in which the system has to survive. A fitness function is used in order to determine the acceptability of a solution with the fittest solutions being carried forward to successive generations. Back (1996) develops alternatives to the traditional genome representation methods (binary string) introducing other methods of representation, such as graphs, trees, and linked lists. In the MSPP the genome will represent a path from the source to destination.

Goldberg (1989) highlights that the mechanics of a simple Genetic Algorithm are surprisingly simple and involve nothing more complex than the copying and swapping of partial strings. Each generation creates a new, slightly modified set of the

old using the ‘fittest’ elements of the previous generation. Periodically the algorithm seeks to increase the coverage of the search space in the form of a mutation operation on a random member of the population set. Figure 3.1 taken from Hoitkotter and Beasley (2000) outlines the typical operation of the Genetic Algorithm computation process. Raidl (2005) suggests that evolutionary algorithms have several advantages over other optimisation techniques:

- Simple Evolutionary Algorithms (EA) do not require any in-depth mathematical understanding of the problems to which they are applied
- Consequently, such EAs are relatively cheap and quick to implement
- EAs are open to problem modifications and can in general cope well with additional constraints, noisy, inaccurate, and incomplete data

Raidl (2005)

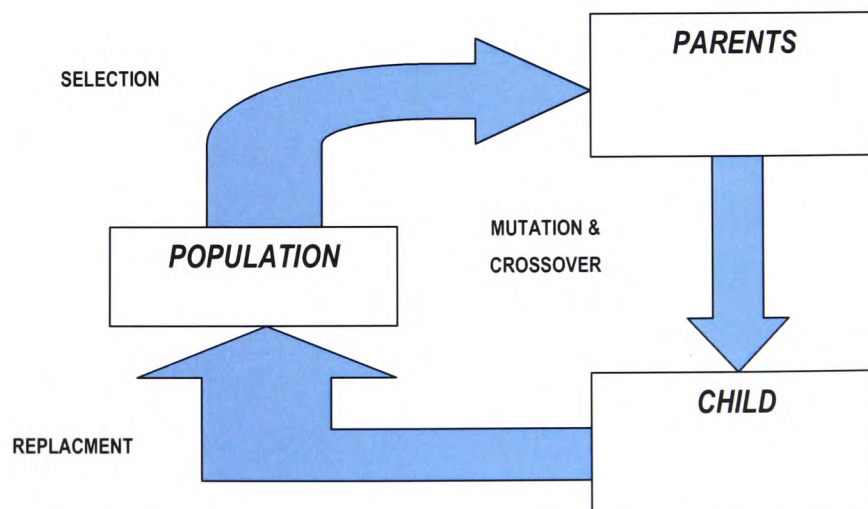


Figure 3.1 Flow of the Basic Genetic Algorithm (Hoitkotter and Beasley, 2000)

Deb (2001) highlights several drawbacks to the use of evolutionary algorithms, the most notable of which are the requirements for effective domain representation and parameter tuning for any given application. Mitchell (1996 p. 167) highlights the notion of premature convergence. If an individual presenting a higher fitness than most of its competitors emerges early on in the course of the run it may reproduce so abundantly that it drives down the population's diversity too soon, leading the algorithm to converge on the local optimum. Schwefel and Rudolph (1998) highlight that traditional algorithms will often outperform genetic approaches in terms of computational efficiency.

Rechenberg (1965) introduced variations to evolutionary algorithms in the form of “evolution strategies” to optimise the processes involved in the design of airfoils. Fogel *et al* (1966) developed a technique entitled “evolutionary programming”. There has been a certain amount of debate in the literature regarding the typology and classification of the various evolutionary approaches in part dependant on the formation members of the chromosomes and therefore the population (Michalewicz and Michalewicz, 1997). That debate largely centres on the representation of genome structures, i.e. if a solution uses a method of representation other than a binary string is the process a Genetic Algorithm or an evolutionary strategy?. The fundamental concepts however remain the same. In this work even though the method of representation is not a binary string the algorithm is considered a Genetic Algorithm.

To date much of the work aimed at the solution of multi objective problems has concentrated on the use of evolutionary/Genetic Algorithms. Mitchell (1996) highlights the use of a population-based structure arguing the technique is able to cover a wide search area simultaneously. Forrest (1993) suggests that Genetic Algorithms are able to produce an optimal result even when sampling only small regions of a search space.

Although based upon relatively simple search techniques the field of evolutionary computation has demonstrated on several occasions that they can be considered proven and reliable methods for solving search and optimisation problems. Oduwaga *et al* (2005 p.293) state “the most significant advantage of using evolutionary search lies in the gain of flexibility and adaptability to the task at hand”. Genetic Algorithms also introduce the concept of genetic operations. The Genetic Algorithm makes use of two main genetic operations in the form of crossover and mutation in order to a) increase genetic diversity and b) mimic the evolutionary approach of ‘survival of the fittest’. Mitchell (1996) argues that the decision as to which genetic operators to use when solving a problem using Genetic Algorithms depends largely on the method of representation. The crossover operator is the most common genetic operator, and involves the selection of two candidate solutions which then divide, swapping components at a given point in order to produce new candidate solutions. The effect is that the child has inherent details from both parents.

Figure 3.2 demonstrates the process of crossover on chromosomes. Spears and DeJong (1990) study multipoint and uniform crossover operators when applied to Genetic Algorithms, reporting the work of Sysweda (1989) which indicated that the use of multipoint crossover might have advantages over single point under certain circumstances. Spears and DeJong (1990) state that larger population sizes are protected from the “disruption” caused by multipoint crossover. Schaffer and Eshelman (1991) empirically compare mutation and crossover, and conclude that mutation alone is not always sufficient. The mutation operator involves the selection of a given candidate solution and attempts to introduce a small random change to that candidate solution. As an example, in a binary string representation of a candidate solution the mutation operator may simply take a random bit and invert it. The approach is highlighted in Figure 3.3. Luger (2002) states that mutation is a key genetic operator as the initial candidate solutions generated may otherwise exclude an essential component of the solution. Mitchell (1996) agrees and highlights that many early forms of evolutionary computation techniques offered only the mutation operator and that the notion of the crossover operator was often absent. Having considered the basic terminology, form and

function of the Genetic Algorithm attention now turns onto the multi objective implementations.

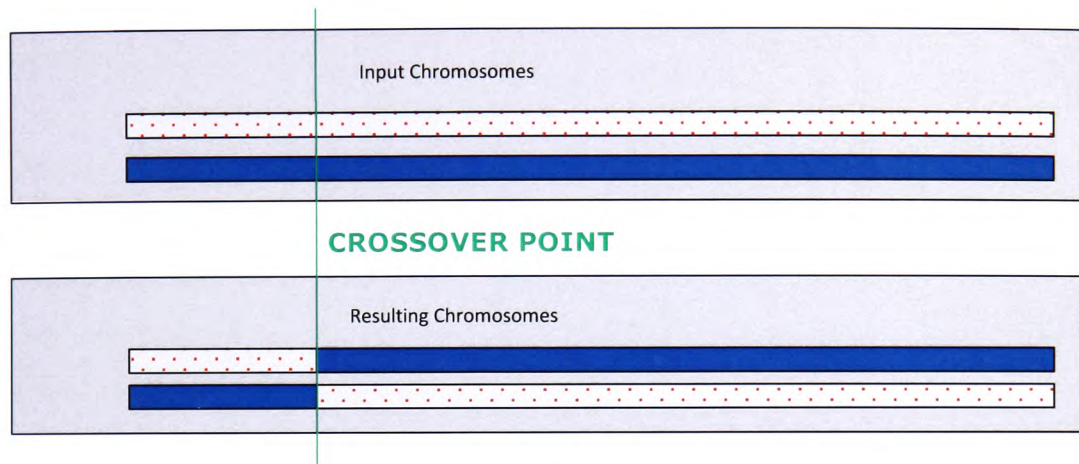


Figure 3.2 Outline of General Crossover Procedure

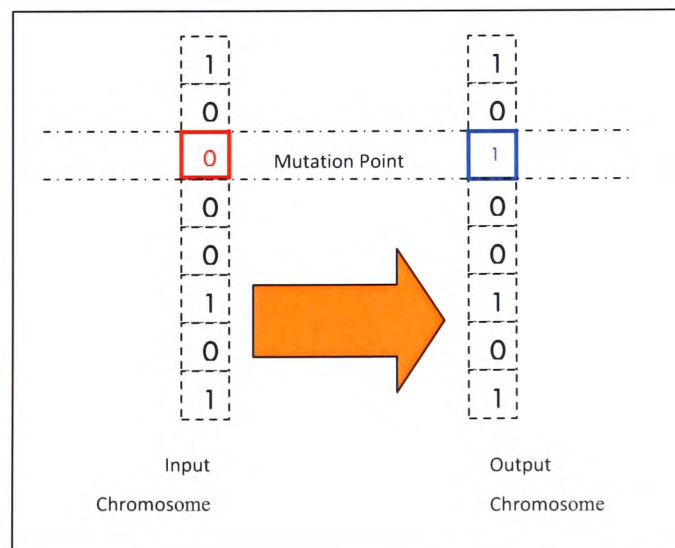


Figure 3.3 Outline of Mutation Operator

3.1.1. Non-Pareto Approaches

Amongst these early methods is the VEGA (Vector Evaluated Genetic Algorithm) of Schaffer (1985). Figure 3.4 presents a general three criteria schematic of the VEGA approach to solving multi objective problems. The VEGA approach consists of a simple Genetic Algorithm with a modified selection mechanism. At each generation a number of sub-populations are generated by performing proportional selection according to each objective function in turn. Assuming the problem consists of C objectives to be optimised, the approach of Schaffer is to split the general population into C sub populations. The process applies a unique fitness function to each subpopulation optimizing each according to one of the criteria under consideration. A process of recombination then takes place in order to form a new general population. Coello-Coello (2000) suggests that the VEGA algorithm does come close to the production of non-dominated (Pareto) solutions but also highlights several criticisms of the approach, the most notable of which relates to its inability to retain solutions with acceptable performance, perhaps above average, but not outstanding for any of the objective functions. These solutions may have been good candidates for becoming non-dominated solutions but could not survive under the selection scheme of this approach. At any generation solutions that can be considered 'good' in all criteria may be discarded because that solution is not the best in any one criterion. Tamaki *et al* (1995) introduce a variation to the VEGA algorithm where at each generation non-dominated solutions are automatically carried over to the next generation. Deb (2001) highlights that the VEGA approach will produce solutions that are good for individual criteria. Coello-Coello (2000) states that the solution set returned by the algorithm achieve what is described as "non-dominated in a local sense".

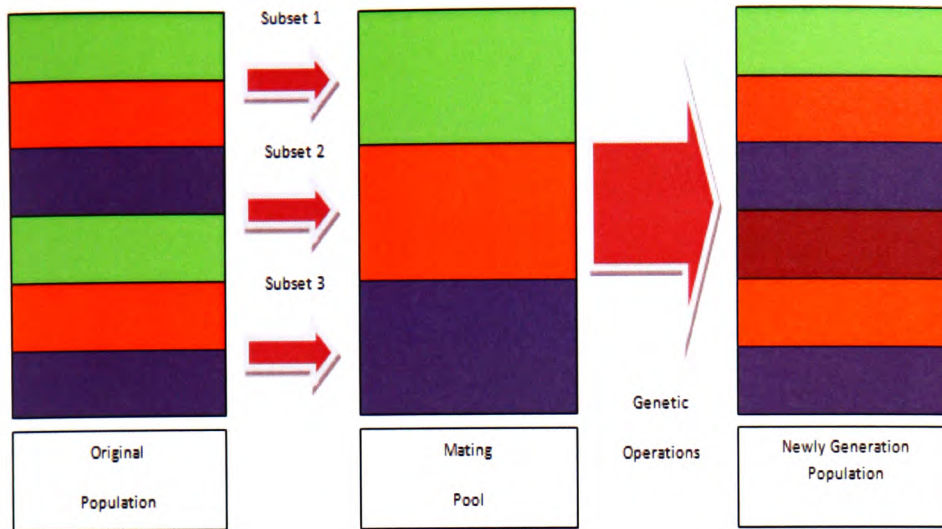


Figure 3.4 Outline of VEGA Algorithm

3.1.2. Population Based Approaches

Goldberg (1989) proposes the use of the Pareto based fitness ranking to assist in the solution of the problems that are present in the work of Schaffer (1985) as discussed in the previous section. The method favours those solutions that are non-dominated with respect to the current generation. Algorithm 3.1 provides a general overview of the ranking mechanism. Having a record of the number of solutions (N) in the current population. The algorithm sets a local variable $CURRENT_RANK = 1$. The process then checks the current population for non dominated solutions. These are assigned the value of $CURRENT_RANK$. Following an initial scan of the population any solutions with the rank of $CURRENT_RANK$ are removed from the population and the value of $CURRENT_RANK$ incremented. The process continues until all members of the population have been scanned. Figure 3.5 presents a view of an example population following the ranking operation. Srinivas and Deb (1995) proposed the Non-dominated Sorting Genetic Algorithm (NSGA). The algorithm makes extensive use of the ranking methodology suggested by Goldberg (1989). Solutions with a higher rank are more likely to be selected as candidates for the genetic operators and to be carried into successive generations.

In Fonseca and Fleming (1993) an alternative ranking scheme to that of Goldberg (1989) is proposed. Fonseca and Fleming's MOGA (Multi-Objective Genetic Algorithm) approach ranks each member of the population in relation to the number of solutions it dominates although. Like the ranking method of Goldberg (1989) the method may be considered a simple extension of the single criteria Genetic Algorithm. The rank of solution(x) is given by Equation 3.1 where r_u^t is the number of solutions dominating solution(x) in a approximation set $PfTRUE$.

$$Rank(x) = r_u^t$$

Equation 3.1 Fonseca and Fleming Ranking Equation

Algorithm:	Goldberg Ranking Method
Input:	S = Set Of Solutions To Be Checked For Ranking
Output:	$R[Solution, Rank]$ = Set Of Solutions With Ranking Information
R = Empty Set $CURRENT_RANK = 1$ $COUNT = S $ WHILE ($ S > 0$) { FOR ($i = 0$ TO $COUNT$) { FOR ($j = 0$ TO $COUNT$) { IF ($i \neq j$) { IF ($S[i]$ Is Dominated By $S[j]$) Mark $S[i]$ As Dominated } } } FOR ($k = 0$ TO $COUNT$) { IF ($S[k]$ Not Marked As Dominated) { $R = R + \{S[k], CURRENT_RANK\}$ Remove $S[k]$ from S } } $CURRENT_RANK = CURRENT_RANK + 1$ $COUNT = S $ } 	

Algorithm 3.1 Goldberg Ranking Method

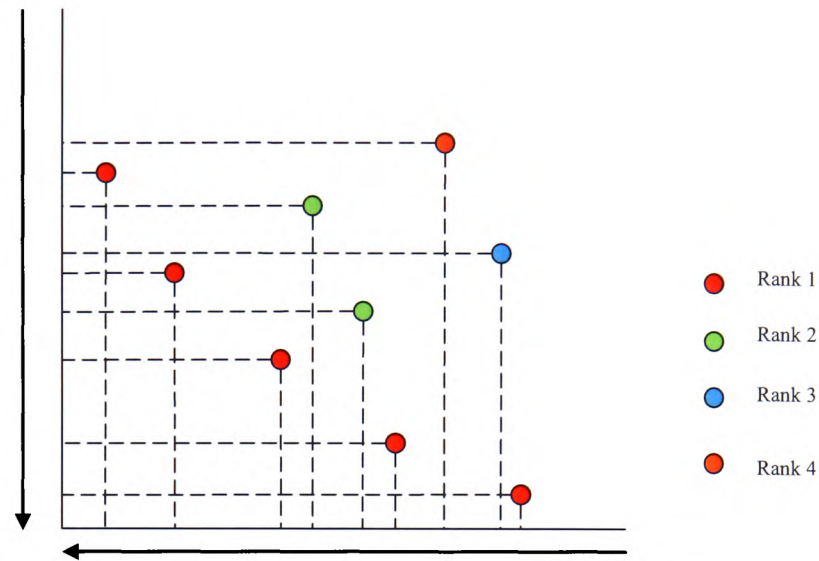


Figure 3.5 Visualisation of the Goldberg Ranking Methodology

Horn *et al* (1994) introduce the concept of nicheing in the '*Niched Pareto Genetic Algorithm*' (NPGA). In this approach the size neighbourhood (or niche) is controlled through the σ share value. A count is made as to how many solutions are located within the diameter of that σ share value and the fitness value decreased proportionally to the number of individuals sharing the same neighbourhood. This aims to promote the generation of solutions in the least populated regions of the search space. The general concept of nicheing is given in Figure 3.6. Two solutions are presented both of which are indifferent with respect to each other. *Solution A* however is in a much more 'crowded' area of the optimal set of solutions, and so under the nicheing scheme of Horn *et al* (1994) would be assigned a lower fitness than *solution B* thereby encouraging diversity of solutions across the front. Difficulties arise in specifying an optimal size of the σ share parameter. Algorithm 3.2 provides an overview of the NPGA. In Algorithm 3.2 two aspects are of particular interest those being the line '*Specialized Binary Tournament Selection*' and '*Return Candidate with lower niche count*'. The latter uses the methodology described above to increase the diversity and spread of solutions across the front. In the '*Specialised Binary Tournament*' two individuals are chosen at random from the population. Those two solutions are then compared against a subset from the entire population. If one of the

two candidate solutions is dominated by the subset of the population and the other is not then the non-dominated individual wins. Where the two solutions are indifferent the result of the tournament is decided through fitness sharing. If Figure 3.6 were selected as an example in the event of a tie solution B would be selected given its lower niche value. The specialised binary tournament scheme is provided in Algorithm 3.3.

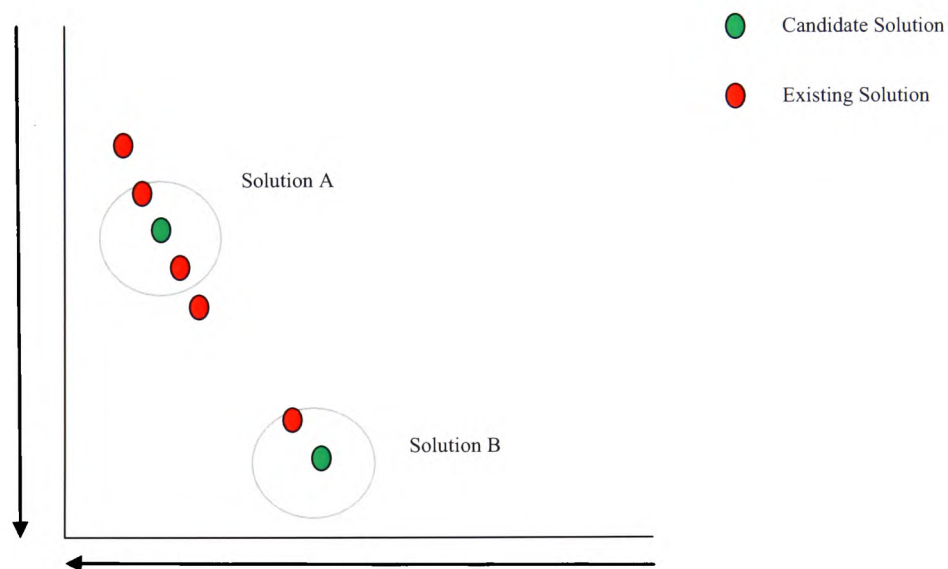


Figure 3.6 Crowd Based Nicheing Example

Algorithm:	NPGA Algorithm
Input:	P = Population Of Candidate Solutions G = Number Of Generations To Perform
Output:	$PfAPPROX$ = Set Of Pareto Approximations
Initialize Population P $COUNT = P $ FOR ($i = 0$ TO $COUNT$) Evaluate Objective Values FOR ($j = 0$ TO G) { Perform Specialized Binary Tournament Selection Returning Solutions A and B and P^{Sub} IF (A Is Dominated With Respect To P^{Sub}) Select B IF (B Is Dominated With Respect To P^{Sub}) Select A IF (A && B Are Dominated With Respect To P^{Sub}) (A && B Are Indifferent With Respect To P^{Sub}) { Perform Specialized Fitness Sharing Return Candidate With Lower Niche Count } Perform Single Point Crossover Perform Mutation FOR ($k = COUNT$) Evaluate Objective Values }	

Algorithm 3.2 NPGA Outline

Algorithm:	NPGA Specialised Binary Tournament Selection
Input:	P = Population Of Candidate Solutions $SUBSETNUMBER$ = Size Of Comparison Subset
Output:	P^{Sub} = A Subset Of P A = Candidate Solution From P B = Candidate Solution From $P \neq A$
<hr/>	
$P^{Sub} = \{ \}$	
WHILE ($ P^{Sub} < SUBSETNUMBER$)	
{	
X = Select Solution From P At Random	
WHILE (P^{Sub} Contains X)	
X = Select Solution From P At Random	
$P^{Sub} = P^{Sub} + X$	
}	
A = Select A Solution From P Not In P^{Sub}	
B = Select A Solution From P Not In P^{Sub} && $\neq A$	

Algorithm 3.3 NPGA Specialised Binary Tournament Selection

3.1.3. Elitist Based Methodologies

Earlier MOEAs such as the MOGA, NSGA, and the NPGA highlighted in the previous section can be criticized due to their simplistic handling of multiple criteria and lack of true methods for handling elitism. The following methods counter these criticisms through the use of methods such as external archiving of non-dominated solutions and methods to increase the coverage of the search such as the introduction of crowd density control functions. The discussion leads to the introduction of

Figure 3.7 which highlights the general properties of each of the three techniques discussed.

Knowles and Corne (1999) introduce the Pareto Archiving Evolutionary Strategy (PAES) algorithm that uses a 1+1 evolution strategy in conjunction with an external solution set that records the non-dominated solutions found. In addition the

algorithm uses an adaptive grid in order to maintain the diversity of the solutions. Algorithm 3.4 highlights the running of the PAES algorithm. The technique mutates a single parent in order to create a single offspring. The mutation process is applied to the parent with a direct comparison between the costs or fitness of the child and the parent performed. If following dominance checks the mutated offspring is found to be a ‘better’ solution than the parent a swap takes place with the mutated solution being accepted as the current solution and a copy of the solution placed into an external archive. Any subsequent mutation is performed on this new parent solution. The use of an external archive allows for the storage of the better solutions discovered to date, the contents of which represent the approximation of the optimal front (*PfTRUE*). The technique may allow indifferent solutions to enter the search process with the provision that those solutions indicate a move to a less crowded area of the search space.

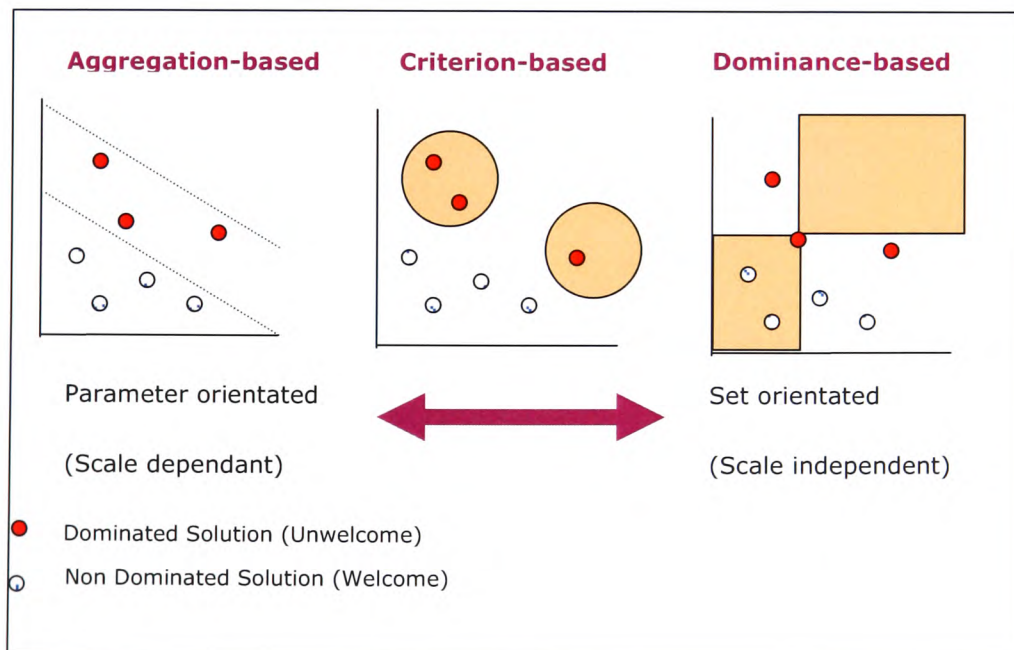


Figure 3.7 Multi Objective Processing Types

A key component of the PAES approach is the use of a crowding procedure based on recursively dividing the objective space in order to determine less populated areas. Knowles and Corne (1999) highlight that the procedure is designed to have two advantages over the nicheing methods used in some multi-objective GAs: Its computational cost is lower and requires little work to determine the parameters used. When each solution is generated its grid location in the solution space is determined. Assuming the range of the space is defined in each objective the required grid location can be found by repeatedly bisecting the range in each objective and finding in which half the solution can be found. The location of the solution is recorded as a binary string of $2^{(ld)}$ where l is the number of bi-sections of the space carried out for each objective, and d is the number of criterion. Each time the solution is found to be in the larger half of the prevailing bisection of the space the corresponding bit in the binary string is set. A map of the grid is also maintained indicating for each grid location how many and which solutions in the archive currently reside there. The number of solutions present in a grid location is referred to as its population. With a maximum archive size of 100, for example, and a two-objective problem with $l=5$, the solution space is divided into 1024 squares. However, the archive is clustered into a small region of this space representing the slowly advancing approximation to the Pareto front. Knowles and Corne (1999) suggest an l value of between three and six provides good results. They also provide $(2 \cdot d \cdot n + n \cdot l \cdot d)$ as the number of operations to update an archive and $(a \cdot l \cdot d)$ operations to find the location of solution in the archive. Algorithm 3.5 provides a high level overview of the archive management functions.

Algorithm:	Basic PAES Algorithm
Input:	<i>ITERATIONS</i> = Number Of Iterations To Perform <i>S</i> = Maximum Size Of External Archive
Output:	<i>PfAPPROX</i> = Set Of Pareto Approximations
<i>PfAPPROX</i> = Empty Set Of Optimal Solutions <i>C</i> = Generate Initial Solution Evaluate <i>C</i>	
FOR (<i>i</i> = 0 TO <i>ITERATIONS</i>)	
{	
<i>M</i> = Introduce Mutation To Solution <i>C</i> Evaluate <i>M</i>	
IF (<i>C</i> Dominates <i>M</i>)	
Discard <i>M</i>	
ELSE IF (<i>M</i> Dominates <i>C</i>)	
{	
<i>C</i> = <i>M</i>	
<i>PfApprox</i> = <i>PfApprox</i> + <i>M</i>	
}	
ELSE IF (<i>M</i> Is Domiated By Any Member Of <i>PfAPPROX</i>)	
Discard <i>M</i>	
ELSE	
Perform TestArchive(<i>C</i> , <i>M</i> , <i>PfAPPROX</i>) To Determine <i>C</i> And <i>PfAPPROX</i>	
}	

Algorithm 3.4 Outline of the PAES Algorithm

Algorithm:	PAES TestArchive Procedure
Input:	<i>PfAPPROX</i> = Set Of Optimal Solutions <i>C</i> = Current Optimal Solution <i>M</i> = Mutated Solution <i>MS</i> = Maximize Size Of <i>PfApprox</i>
Output:	<i>PfAPPROX'</i> = Updated Set Of Pareto Approximations <i>C'</i> = Updated Current Solution // <i>C</i> or <i>M</i>
<pre> IF (<i> PfAPPROX </i> < <i>MS</i>) { <i>PfAPPROX'</i> = <i>PfAPPROX</i> + <i>M</i> IF (<i>M</i> Is In A Less Crowded Region Of <i>PfAPPROX</i>) <i>C'</i> = <i>M</i> ELSE <i>C'</i> = <i>C</i> } ELSE { Identify Most Crowded Area Of <i>PfAPPROX</i> IF (<i>M</i> Would Occupy Less Crowded Area) { Remove A Solution From Most Crowded Area Of <i>PfAPPROX'</i> <i>PfAPPROX'</i> = <i>PfAPPROX</i> + <i>M</i> IF (<i>M</i> Is In A Less Crowded Area Of <i>PfAPPROX'</i> Than <i>C</i>) <i>C'</i> = <i>M</i> ELSE <i>C'</i> = <i>C</i> } ELSE { IF (<i>M</i> Is In A Less Crowded Area Of <i>PfAPPROX'</i> Than <i>C</i>) <i>C'</i> = <i>M</i> ELSE <i>C'</i> = <i>C</i> } } </pre>	

Algorithm 3.5 PAES Test Archive Procedure

The Strength Pareto Evolutionary Algorithm (SPEA) proposed by Zitzler and Thiele (1999) is a simple and effective Genetic Algorithm that ranks solutions purely on dominance. An external archive of non-dominated solutions is maintained and a clustering method ensures that the archive does not grow larger than a predefined limit while maintaining the diversity within the archive. Parents for reproduction are selected from the union of the previous offspring and the archive through binary tournament selection with replacement where the quality of solutions is ranked by the proportion of solutions that they dominate or are dominated by. SPEA is an effective algorithm for multi-objective optimisation. However given its simplistic nature several enhancements can be seen in the literature.

Fieldsend *et al* (2001) introduce a number of extensions to the SPEA algorithm such as the maintenance of all non-dominated solutions discovered in the main population. The historical set of non dominated solutions become an active input to the search process. Zitzler *et al* (2001) introduces the SPEA2 algorithm. An update to the original algorithm that aims to resolve some of the perceived key failings of the SPEA algorithm that could under certain circumstances lose outer solutions. The SPEA2 approach like the PAES algorithm of Knowles and Corne (1999) aims to maintain these solutions with the aim of ensuring that “a good spread of non-dominated solutions” (Zitzler *et al*, 2001 p.5) is maintained. The SPEA2 approach presents a modified elitist archive which is no longer purely elitist but which is made up of a fixed number of solutions. In those cases where there is a shortage of solutions the archive is filled with dominated solutions. Where the size of the archive exceeds that specified as a maximum a clustering methodology ensures an even spread of solutions across the Pareto front. A final change between the two SPEA approaches can be seen in that the SPEA2 approach now limits the selection mechanism to the external archive of solutions. Algorithm 3.6 and Algorithm 3.7 demonstrates the development of the original SPEA algorithm into the SPEA2 algorithm.

Algorithm:	SPEA Algorithm
Input:	P = Size Of Population G = Number Of Generations MPS = Size Of Mating Pool MS = Maximum Number Of Optimal Solutions
Output:	$PfAPPROX$ = Set Of Pareto Approximations
<hr/> <i>POPULATION</i> = Generate Random Initial Population Sized P <i>PfAPPROX</i> = Empty Set Of Pareto Optimal Solutions <i>MP</i> = Mating Pool = { } <hr/> FOR ($j = 0$ TO G) { FOR ($i = 0$ TO P) Evaluate Objective Function Of <i>POPULATION</i> [i] <i>PfAPPROX</i> = Extract Non Dominated Solutions From <i>POPULATION</i> IF ($ PfAPPROX \geq MS$) Prune <i>PfAPPROX</i> Using Clustering FOR ($i = 0$ TO P) Evaluate Objective Function Of <i>POPULATION</i> [i] FOR ($i = 0$ TO $ PfAPPROX $) Evaluate Objective Function Of <i>PfAPPROX</i> [i] WHILE ($ MP \leq MPS$) Use binary tournament selection with replacement to select Solutions From <i>POPULATION</i> + <i>PfAPPROX</i> adding candidates to <i>MP</i> <i>POPULATION</i> = New Population Produced Using Mutation And Crossover From <i>MP</i> }	

Algorithm 3.6 SPEA Algorithm

Algorithm:	SPEA2 Algorithm
Input:	P = Size Of Population G = Number Of Generations MPS = Size Of Mating Pool MS = Maximum Number Of Optimal Solutions
Output:	$PfAPPROX$ = Set Of Pareto Approximations
$POPULATION$ = Generate Random Initial Population Sized P $PfAPPROX$ = Empty Set Of Pareto Optimal Solutions MP = Mating Pool = { }	
FOR ($j = 0$ TO G) { FOR ($i = 0$ TO P) Evaluate Objective Function Of $POPULATION[i]$ $PfAPPROX$ = Extract Non Dominated Solutions From $POPULATION$ IF ($ PfAPPROX \geq MS$) Prune $PfAPPROX$ Using Truncation ELSE { WHILE ($ PfAPPROX \leq MS$) Copy A Dominated Solution From $POPULATION$ To $PfAPPROX$ } } FOR ($i = 0$ TO P) Evaluate Objective Function Of $POPULATION[i]$ FOR ($i = 0$ TO $ PfAPPROX $) Evaluate Objective Function Of $PfAPPROX[i]$ WHILE ($ MP \leq MPS$) Use binary tournament selection with replacement to select Solutions From $POPULATION$ + $PfAPPROX$ adding candidates to MP $POPULATION$ = New Population Produced Using Mutation And Crossover From MP } 	

Algorithm 3.7 SPEA2 Algorithm

3.2. Multi Objective Simulated Annealing

Simulated Annealing is a generalization of a Monte Carlo method for “examining the equations of state and frozen states of n-body systems” (Metropolis *et al*, 1953). In an annealing process a solid is first raised to a given temperature and then slowly cooled over time. At higher temperature the atoms making up the solid form a ‘chaotic’ state with random jumps between various states being made. As the temperature lowers and the state of the solid becomes more ordered the atoms making up a solid become much more “controlled” and the ability to move limited. In software form the temperature controls the ability to move to less optimal solutions, thus introducing the ability to escape from local optima. Single criterion Simulated Annealing has been widely applied to a number of applications including structural optimisation (Kolahan *et al*, 2007), map generalization (Ware *et al*, 2003) and labelling (Christensen *et al* 1995). Geman and Geman (1984) provide a proof that the method will when allowed sufficient time achieve a global optimum. Prior to introducing the multi objective approaches to Simulated Annealing seen in the literature the approach is first considered in single criterion form.

The Simulated Annealing algorithm is initialized through the generation of a random solution and by setting the temperature parameter T . Then the following is repeated until the termination condition is satisfied: A solution s' from the neighbourhood $N(s)$ of the solution s is randomly sampled. In effect s' is s with a change introduced. It is accepted as the new current solution depending on the fitness(f) or cost of the solutions $f(s)$ and $f(s')$ and T . s' replaces s if $f(s') < f(s)$ or, in cases where $f(s') \geq f(s)$ with a probability which is a function of T and $f(s') - f(s)$. The probability is generally computed following the Boltzmann distribution given in Equation 3.2. The terminating condition is usually specified as a small value such as 0.00001. The temperature T is decreased during the search process. Thus at the beginning of the search the probability of accepting ‘inferior’ moves is high yet over time the probability of accepting inferior solutions decreases as the value of T decreases. A basic implementation of the approach is given in Algorithm 3.8.

$$P = \exp(-(f(s') - f(s))/T)$$

Equation 3.2 Boltzmann Distribution

Where $f(s)$ is the fitness of the current solution

And $f(s')$ is the fitness of the mutated solution

And T is the current annealing temperature

And P , the result is the probability of accepting an inferior solution

Algorithm:	Basic Simulated Annealing Search Algorithm
Input:	T_0 = Starting Temperature T_C = Closing Temperature T_CR = Annealing Cooling Rate
Output:	S_{Best} = Optimal Solution Discovered // A Form Of Elitism
S = Generate Initial Solution $S_{Best} = S$ $T = T_0$ WHILE ($T \geq T_C$) { $S' =$ Select Alternative Solution From Neighbourhood Of S IF ($f(S') < f(S)$) $S = S'$ ELSE $S = S'$ With Probability $p(T, \text{Fitness}(S'), \text{Fitness}(S))$ IF ($f(S) < f(S_{Best})$) $S_{Best} = S$ $T = T * T_CR$ }	

Algorithm 3.8 Basic Outline of Simulated Annealing

The concept of neighbourhood is important in a number of heuristic approaches and a key, problem-specific choice concerns the neighbourhood function definition. The efficiency of Simulated Annealing is highly influenced by the neighbourhood function used (Moscato, 1993). For instance, in the travelling salesman problem the neighbourhood is often considered the pair-wise swapping of any two city locations. Alizamir *et al* (2009 p.15) suggest: “roughly speaking, a more complicated neighborhood structure may cover a wide range of the feasible region and has the potential of moving far away in a few number of iterations while a simple neighborhood structure needs far more iterations to move from one part of feasible region to another”. Goldstein and Waterman (1988 p. 411) largely concur and in addition state: “The question now arises: what choice of neighborhoods N , will allow the algorithm to converge quickly? Intuitively, it **seems** that a neighborhood system that strikes a compromise between these extremes would be best”. Alizamir *et al* (2009 p.4) proceed to list several criteria for effective neighbourhood selection:

Effectiveness: the power of the neighborhood structure in covering the whole feasible space.

Efficiency: the efficiency of a neighborhood structure which is the quality of its performance in covering the feasible region depends on several (contradictory) factors:

Speed: the number of moves needed to reach any arbitrary point in the feasible region

Computational Effort: the computations needed for each movement.

Size (Number of Neighbors): the size of a neighborhood structure is defined as the number of solutions which are accessible in an immediate move from the current solution. A larger number is usually an advantage as any arbitrary solution can be reached in less number of moves.

Information Volume: the amount of information transformed. This information may be used to perform better moves through the feasible space. For instance, there are gradients, Hessian matrix, eigen values and convexity information for the continuous space and taboo list, function characteristics and lower & upper bounds for the discrete space.

Alizamir et al (2009)

The discussion on Simulated Annealing as applied to single criterion algorithms is concluded in Algorithm 3.9. It shows the addition of an inner loop where multiple neighbour solutions are considered at each generation. The best solution from the neighbours obtained in the inner loop is recorded. Following the termination of the inner loop the algorithm continues as normal.

Algorithm:	Simulated Annealing Search Algorithm With Multiple Neighbours
Input:	T_0 = Starting Temperature T_C = Closing Temperature T_{CR} = Annealing Cooling Rate SN = Size Of Neighbourhood
Output:	S_{Best} = Optimal Solution Discovered // A Form Of Elitism
N = Empty Set Of Solutions From Neighbourhood Of S . S = Generate Initial Solution $S_{Best} = S$ $T = T_0$ WHILE ($T \geq T_C$) { FOR ($i = 0$ TO SN) $N = N + PS'$ // Select Alternative Solution From Neighbourhood Of S $S' =$ Select Fittest Solution From N IF ($f(S') < f(S)$) $S = S'$ ELSE $S = S'$ With Probability $p(T, \text{Fitness}(S'), \text{Fitness}(S))$ IF ($f(S) < f(S_{Best})$) $S_{Best} = S$ $T = T * T_{CR}$ $N = \{ \}$ // Empty The Neighbourhood Set } 	

Algorithm 3.9 Simulated Annealing with Multiple Neighbours

An early attempt at the solution of multi criteria problems using the Simulated Annealing approach can be seen in the work of Serafini (1992) in which a bi-objective methodology is developed. In the approach of Serafini (1992) an initial solution (X) is generated randomly from within the search space. Following the perturb mechanism a solution (X') is generated from within the neighbourhood of the initial solution X . If the solution X' is non-dominated when compared to solution X then the modified solution (X') is added to an external archive of Pareto optimal solutions. That external archive will from now be known as *PfAPPROX* – the approximation of the Pareto front. The *PfAPPROX* set is extracted from this external archive set of solutions when the temperature has reached a terminating value. Serafini (1992) suggests that one of the major considerations is how and when to replicate the annealing process when dealing with multiple criteria. That is to say, how to deal with situations when solution X' is either dominated or indifferent to the solution X . The traditional approach of the Simulated Annealing algorithm would involve the random acceptance of such a mechanism based upon the temperature at a given time together with a comparative fitness value. As the temperature decreases the probability of accepting a dominated or indifferent solution will decrease in line with the temperature. The approach taken by Serafini (1992) is to combine the sum of all criteria into a single metric.

The work of Ulungu *et al* (1999) shares many similarities with that of Serafini (1992), in that both condense the multi objective problem into a single objective problem through aggregation. Ulungu *et al* (1998) present an interactive Simulated Annealing approach where users specify weightings for the considered criteria. The authors also discuss further the concepts of neighbourhood. Ray *et al* (1995) also make use of a weighted sum approach as part of the indifferent or dominated acceptance technique. Czyak and Jaskiewicz (1998) present a hybrid Simulated Annealing and Genetic Algorithm. The algorithm makes use of a “generating set” to assist in the management of weights, which are in turn used as the basis of the acceptability of a given solution based upon indifference or domination. At each iteration multiple assessments are made with various weightings replicating the population factor of the Genetic Algorithm.

The Suppapitnarm *et al* (2000) approach makes use of a ‘composite energy difference’ for the acceptance criteria when reviewing indifferent solutions. Instead of weighting and summing the objectives to produce a composite energy difference for the acceptance criteria this algorithm uses a multiplicative function with individual temperatures for each objective with each weighting adjusted independently by the algorithm. These multiplicative energy functions are equivalent to a weighted sum of logs of the objectives. This removes the need for the assignment of weighting values to any of the objectives prior to the run. It should be noted however that the search process undertaken still limits the output to a single points on the front $PfTRUE$. Suppapitnarm *et al* (2000) also employ a ‘return-to-base’ scheme whereby the current solution is merged with another solution from the non-dominated archive to promote a better coverage of the front $PfTRUE$ (the set of optimal solutions) and further increase the ability to escape local optima.

Suman and Kumar (2006) report on the increasing acceptance of SA for multi criteria analysis and suggest the following properties of Simulated Annealing for that acceptance noting that the methodologies will:

- find multiple solutions in a single run
- work without derivatives
- converge speedily to Pareto-optimal solutions with a high degree of accuracy
- handle both continuous function and combinatorial optimisation problems with ease
- be less susceptible to the shape or continuity of the Pareto front.

Suman and Kumar (2006)

Suman (2005) makes an attempt to reduce the runtime required using the algorithms. Various stopping criteria are highlighted such as specifying the total number of iterations to be performed and a subset of that principle, the number of iterations at each temperature in the annealing process. The author proposes the ‘*FROZEN*’ mechanism. If the move does not find a better solution in a predefined number of iterations it is assumed that the algorithm will not generate further improvement and it is stopped. Suman (2005) highlights Pulido and Coello-Coello (2004) who apply a similar mechanism in evolutionary computation i.e. terminating after a fixed number of iterations with no improvement. Suman (2005, p.1135) states:

“The total number of iterations required to obtain a good approximation of the true Pareto set, depends on many parameters like complexity, nature, feasible solutions, etc. of a problem. These parameters make the selection of total number of iterations, indeed, a difficult task. If less number of iterations is used, the quality of solutions generated in Pareto set will be bad. On the other hand, if an algorithm overruns towards the end no improvement in the quality of solutions is made as no solution has been placed in the Pareto set. But, the computational cost to obtain the solutions has increased. In either ways, it is related to Pareto set generated”.

Smith *et al* (2008) propose the use of an ‘energy measure’ rather than a weight combination of criteria. In that work rather than using an aggregated weight the authors generate a vector between the current solution and the results of the perturb mechanism. Figure 3.8 presents a view of the energy measure value on an example Pareto optimal front.

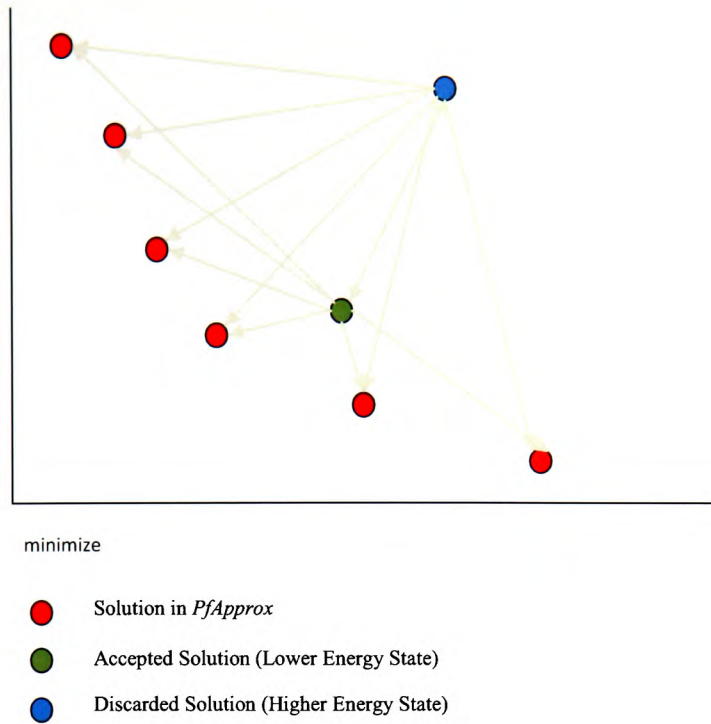


Figure 3.8 Example Energy Measure Determination of Two Solutions

Smith *et al* (2008) report positive results using this method although the size of the front is large in the reported experiments. Bandyopadhyay *et al* (2008 p.270) propose a similar metric of “amount of dominance in order to determine the acceptance of a new solution”. Bandyopadhyay *et al* (2008) suggest the use of multiple probability metrics the use of which depends on the domination relationship between the candidate solutions with indifferent solutions receiving a different acceptance probability to solutions where the mutated solution dominates the current solution. Singh *et al* (2010) highlight that the approaches of Bandyopadhyay *et al* (2008) and Smith *et al* (2008) introduce a complex number of checks performed to determine the dominance relationship between solutions. Singh *et al* proceed to introduce a methodology that extends that of Bandyopadhyay *et al* (2008) to better suit the introduction of constraints to objective values. It is of interest to note that Singh *et al* do not present any resolution to the complex dominance checks they highlight.

In Li and Landa-Silva (2011) an algorithm making use of aspects of both evolutionary algorithms and Simulated Annealing approaches is proposed. The approach combines the selection scheme of the VEGA algorithm (Schaffer, 1985) with the probabilistic acceptance measures of Simulated Annealing. The methodology removes any genetic crossover procedure, replacing it with Simulated Annealing mechanisms. The authors report the works of (Merz, 2000; Krasnogor, 2002 and Hart *et al.*, 2004) who adopt a similar approach when applied in single criterion optimisation. They make use of genetic search to explore the global search space while local search is used to examine locally optimal solution spaces. Nam & Park (2000) define several schemes for calculating the 'energy difference' controlling acceptance similar to Serafini (1994). Based on a small empirical study of two-objective problems they suggest that the best is the average difference in objective values.

Ulbricht (2012) compares single and multi criteria approaches to Simulated Annealing and in addition provides a cross comparison with Genetic Algorithms in the form of the SPEA2 algorithm introduced in this chapter. The Simulated Annealing approach when handling multi objective is based on that of Bandyopadhyay *et al* (2008). The authors highlight a comparison of tests performed on a single objective Genetic Algorithm and Simulated Annealing together with multi objectives based on the SPEA2 based approach. Little difference is seen between single and multi objectives Genetic Algorithm approaches. The experiments undertaken are time based with the authors seeking the optimal results within a given period.

3.3. Multi Objective Tabu Search

The basic concept of Tabu Search as described by Glover (1986 p.541) is "a meta-heuristic superimposed on another heuristic". The technique attempts to enable escape from some local optima via the introduction of a memory function the purpose of which is to prevent the algorithm from revisiting recent solutions and so to try to seek new solutions within the search space. This memory function forms the basis of the 'Tabu' of the method name. The method is actively researched and has been widely

used in a variety of applications. Glover and Laguna (1998) attempt to formalize the nature of various levels of memory structure when applied to the Tabu Search algorithm as assisting in the formation of three distinct phases. These phases are referred to as the preliminary search, the intensification phase and the diversification phase. Figure 3.9 of Jaeggi *et al* (2008) presents an outline of how the memory structure can be considered in terms of the application to multiple criteria problems.

The notion of memory is central to the Tabu Search. In order to improve the efficiency of the search process the technique aims not only to keep track of local information but also the search history. While other heuristic techniques limit the concept of memory to the fitness of the best solution s considered the Tabu Search maintains a historical record of the search process. The role of the memory will be to restrict the choice to some subset of $N(i)$ by forbidding moves to those solutions that have recently been considered. An outline of the single criteria Tabu Search process is given in Algorithm 3.10. The notion of neighbourhood and the considerations for the optimal selection of the neighbourhood as seen in the Simulated Annealing algorithm also apply to the Tabu Search.

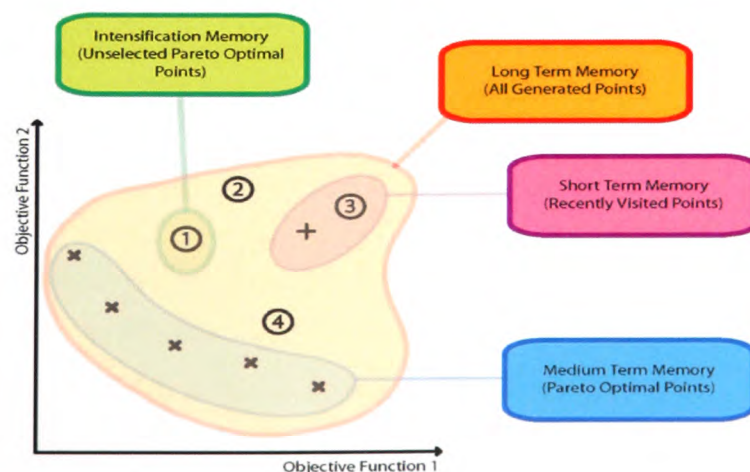


Figure 3.9 Multi Level Memory Model (Jaeggi et al, 2008)

Algorithm:	Tabu Search Algorithm
Input:	<i>ITERATIONS</i> = Number Of Iterations To Perform <i>TABU_SIZE</i> = Size Of The Tabu List <i>SN</i> = Size Of Neighbourhood
Output:	<i>SBest</i> = Optimal Solution Discovered // A Form Of Elitism
<i>N</i> = Empty Set Of Solutions From Neighbourhood Of <i>S</i> . <i>S</i> = Generate Initial Solution <i>SBest</i> = <i>S</i> <i>TABU_LIST</i> = Set Of Solutions Considered Tabu	
WHILE (<i>i</i> = 0 TO <i>ITERATIONS</i>) { FOR (<i>i</i> = 0 TO <i>SN</i>) { <i>PS'</i> = Generate A Solution In The Neighbourhood Of <i>S</i> WHILE (<i>TABU_LIST</i> Contains <i>PS'</i>) <i>PS'</i> = Generate A Solution In The Neighbourhood Of <i>S</i> <i>N</i> = <i>N</i> + <i>PS'</i> // Select Alternative Solution From Neighbourhood Of <i>S</i> } <i>S'</i> = Select Fittest Solution From <i>N</i> IF ($f(S') < f(S)$) <i>S</i> = <i>S'</i> ELSE <i>S</i> = <i>S'</i> With Probability $p(T, f(S'), f(S))$ IF ($f(S) < f(SBest)$) <i>SBest</i> = <i>S</i> Prune <i>TABU_LIST</i> To Size <i>TABU_SIZE</i> <i>i</i> = <i>i</i> + 1 <i>N</i> = { } // Empty The Neighbourhood Set } }	

Algorithm 3.10 Single Criteria Tabu Search

The historically accepted approach taken to multi-criteria optimisation using the Tabu Search shares similarities with that of Simulated Annealing and has again involved the reduction from multi to single objective problem types. Notable works using the Tabu Search approach can be seen in the Multi Objective Tabu Search (MOTS*) approaches of Hansen (1997) and Gandibleux (1997). In the work of Gandibleux (1997) various weighting values are applied to the aggregating function. The algorithm introduces variations to these values independently to increase diversity

in the search process. The work of Hansen (1997) also adopts the approach of applying weightings to give an indication of preference by the user to or against a given objective. Hansen discusses in depth the importance of the neighbourhood to the Tabu Search mechanism. Hansen (1997, p. 10) states: “With a neighborhood function which contains many neighbors for each solution, it can be more efficient to make moves based on a (probabilistic or systematic) sampling of the neighborhood, or in other ways reduce the neighborhood size”. Later Hansen (p.11) suggests, “With neighborhood functions well suited for Tabu Search, however, we may be able to locate the best neighbor without explicitly having to generate all the neighbors”.

The work of Hertz (1994) compares three methods where multiple criteria are condensed into a single aggregated value. The work also introduces a hierarchy structure to the problem where lower “valued” criteria are used merely to indicate a priority where any series of solution may be considered indifferent. Applied to a real world MSPP three criteria are considered; distance, speed and road type. Where distance and speed solutions are indifferent a solution focused more on higher speed roads may be accepted. Jaeggi *et al* (2008) attempt to model the Tabu Search using a Pareto optimal approach rather than to treat the solution of such problems as a variation of an aggregation function. The approach taken here introduces the notion of short, medium and long term phases of memory (periods that solutions remain in Tabu) in order to intensify the search process. In the method suggested in the work medium term memory is represented by an external set of Pareto optimal solutions which form the seeds of a diversification method in the Tabu Search process. Short term memory consists of solutions recently covered. The authors make use of multiple neighbour generation methods and the random selection of indifferent solutions from within the locally non-dominated solutions.

In Kulturel-Konak *et al* (2006) the “Multi nominal Tabu Search” (MTS) is proposed. The method described in that work selects an individual objective to be optimised at each iteration based upon a given probability vector with the authors stating the aim is to “remedy some general obstacles of the classical methods of multi-

objective optimisation (i.e., weighting and scaling of each objective), while maintaining computational ease” (Kulturel-Konak *et al*, 2006 p.930). In order to assist in this the authors suggest several mechanisms including integrated constraints based acceptance or denial of solutions and the dynamic resizing of the Tabu list based upon the history of activity of a archive set of non-dominated solutions. Terano *et al* (2006) produce a hybrid Genetic Algorithm approach that combines a Tabu Search and Genetic Algorithm. They implement two tabu lists; one (long-term memory) represents the best solutions discovered during the run. While the short-term memory stores the optimal solutions discovered in a predefined number of iterations. Jaffrès-Runser *et al* (2008, p.3900) directly compare aggregated function approaches to solving multi objective problems using the Tabu Search with multi objective approaches for designing wireless networks. They find that “In terms of computational time, the mono-objective search performs far better than the MO approach but the tuning of the mono-objective evaluation function parameters takes several launches to get the desired trade-off”.

Grandinetti *et al* (2012) present an algorithm that is of interest due to its production of the Pareto optimal set and making use of Tabu Search. The work makes use of two phase approach. In the first phase a Tabu Search approach is used to generate candidate paths for the vehicle routing problem. As a second and separate stage the paths generated are subjected to analysis using multiple criteria for the production of the optimal front(s). The Tabu Search mechanism, based upon that of Brandao and Eglese (2008) is separate from that of the multi objective approach. The Brandao and Egelese approach performs a combination of several values into a single metric. Grandinetti *et al* (2012) highlight from the perspective of their work the important consideration is the method used to generate possible paths. Tabu search has commonly been used for the solution vehicle routing problems. A search of the literature reveals a large number of works including those of Cordeau and Maischberger (2012), Brandao and Mercer (2012) and Escobar *et al* (2013) that meet three distinct requirements, firstly solving routing problems on graphs, secondly handling multiple objectives and finally making use of Tabu Search. However, all three and others reviewed either combined all objectives into a single value or handled each objective separately in a

linear algorithm and considering certain objectives as constraints or limitation of service requests. Zhiping and Yuxing (2010) develop an approach based upon a parallel implementation of the Tabu Search. Their algorithm applies multi weights to criteria costs where rather than specific weights are select ranges for the weights are selected. The tabu process is then applied in parallel with the merger of all sets and the extraction of the optimal set.

3.4. Quality Measurement in Multi Criteria Based Optimisation

One of the principle difficulties of any experimentation performed on multi objective search problems can be seen in how to measure the completeness of the algorithms developed. Previous sections of this work (*see Chapter 2*) have sought to summarize how traditional approaches to path planning problems have been dealt with purely in terms of efficiency or effectiveness i.e. how effective is a particular data structure in decreasing the run time of algorithms such as the Dijkstra shortest path algorithm? If the problem considered is one where multiple objectives are to be evaluated then it is likely that there will be no single optimal solution or the alternative of an ordered set of solutions. Such problems are considered “comparative” problems. As there is no single ‘best’ solution to the question being asked the ideal answer will vary on any number of criteria. Zitzler *et al* (2003 p.2) suggest, “The notion of performance includes both the quality of the outcomes as well as the computational resources needed to generate this outcome”. However, in order to effectively compare multi objective approaches such metrics do have substantial value.

Veldhuizen and Lamont (1999), Shaw *et al* (1999) and Ripon *et al* (2007) highlight several reasons why the definition of quality becomes much more difficult when dealing with problems that are multi criteria in their nature. In those works it is stated that when dealing with criteria that may often be competing with each other the use of a single metric is unlikely to result in an effective analysis. One of the historic approaches of undertaking an analysis of multi objective problems has been based upon

a visual inspection of the sets *PfTRUE* (the known complete answer) in conjunction with the output of any algorithms developed to add in the solution of the problems (*PfAPPROX*). When faced with a simple bi-criteria problem the spread and proximity of solutions to the Pareto front can be inspected on a simple line/scatter graph representation. A similar situation exists when the number of criteria is increased to three. Going beyond three dimensions/criteria increases the difficulty posed in undertaking an analysis.

Since the work of Veldhuizen and Lamont (1999) several other detailed studies have been undertaken into the problem of assessing the quality of such solutions. However the value that remains in the simple yet effective use of visualisation as an analysis technique into the quality of solutions should not be underestimated. Zitzler *et al* (2004, p.29) suggests, “That in general the quality of an approximation set cannot be completely described by a finite set of distinct criteria such as distance and diversity”. The works of Deb (1999) and Shaw *et al* (1999) highlight that in order to reach an effective analysis of a multi objective problem it is necessary to measure factors such as spacing, diversity, and general search space coverage. The metrics used in the literature to assist in the solution of multi objective problems fall into two distinct categories; those that may be considered purely quantitative metrics and those which attempt to measure the ‘quality’ of the solution provided by the algorithm. It should become apparent when reading the following section however that a great deal of importance is placed upon having some pre-acquired knowledge of the front *PfTRUE*. The ideal (or close to it) answer should have been developed in order to judge the quality of the output from the algorithms. Figure 3.10 suggests an example of a front of *PfTRUE* and *PfAPPROX*. Table 3.1 gives the member variables for those two fronts. Both functions F1 and F2 are examples of minimization problems. When discussing quality measures frequent reference is made to both Figure 3.10 and Table 3.1.

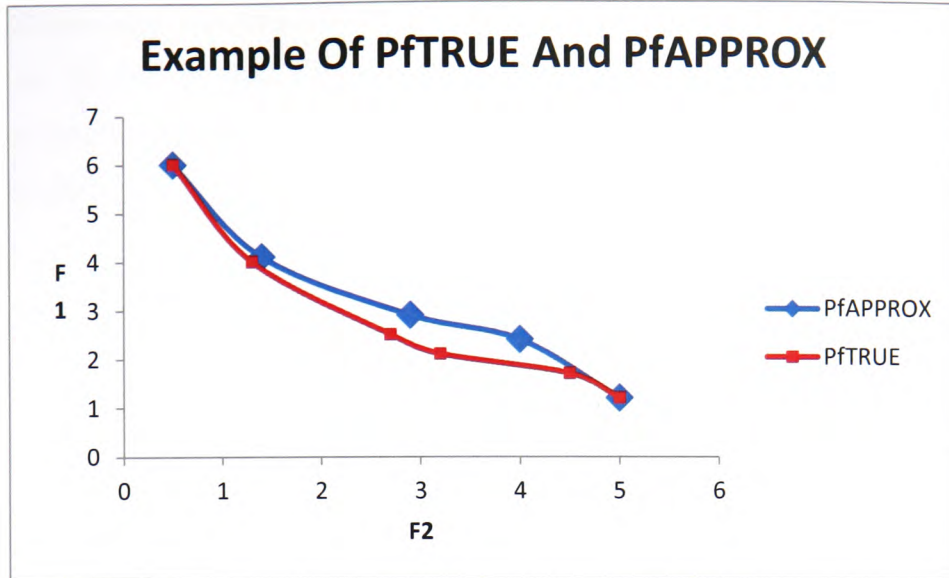


Figure 3.10 Example of $PfTRUE$ and $PfAPPROX$

$PfTRUE$		$PfAPPROX$	
F2	F1	F2	F1
5	1.2	5	1.2
4.5	1.7	4	2.4
2.7	2.5	2.9	2.9
2.7	2.5	1.4	4.1
1.3	4	0.5	6
0.5	6		

Table 3.1 Comparison of Sets $PfTRUE$ and $PfAPPROX$ Based on Figure 3.10

3.4.1. Convergence Measuring Metrics

The simplest performance metric is a simple count of the number of solutions that form a particular front. In the case of the example presented in Figure 3.10 and Table 3.1 the values would be 6 for the front $PfTRUE$ and 5 for the front $PfAPPROX$. The simplistic nature of the metric however gives no insight in the quality of the solution

returned from any algorithm. The simple summation of the member(s) of any front leads onto the Overall Non-dominated Vector Ratio (ONVGR) metric that returns the ratio between the solutions known to be in $PfTRUE$ and those discovered in $PfAPPROX$ (Van Veldhuizen & Lamont, 1999).

$$ONVGR = \frac{|PfAPPROX|}{|PfTRUE|}$$

Equation 3.3 ONVGR Metric

Where $PfTRUE$ is the set of optimal solutions previously calculated.

And $PfAPPROX$ is the of solutions generation representing the approximation of $PfTRUE$

In the case of the solution sets given in the examples provided (Figure 3.10 and Table 3.1) the ONVGR returned from any analysis would be 0.83 (5/6). Again, the ONVGR metric is simplistic in nature and adds little in the way of true meaning. A returned value of one indicates that in terms of quantity $PfTRUE$ and $PfAPPROX$ are equal. However, this does not indicate that the two fronts are equal using other factors merely that they contain the same number of points and so do not really reflect the ‘quality’ of the solution and is therefore of limited value when used as the sole quality metric.

Where the ONVGR metric attempts to measure the relationship that exists between the contents of the front $PfTRUE$ and $PfAPPROX$ the error ratio (ER) metric (Van Veldhuizen, 1997) measures the number of solutions in $PfAPPROX$ that are not present in $PfTRUE$.

$$ER = \frac{\sum_{i=1}^n e_i}{n}$$

Equation 3.4 Error Ratio Metric

Where n is $|PfAPPROX|$

And $e_i = 0$ if the solution is an element of $PfTRUE$ or $= 1$ if the solution is not a member of the set $PfTRUE$.

The metric obtained from the ER equation should be between zero and one. The closer the value is to zero then the closer to the front $PfTRUE$ the set $PfAPPROX$ is. Table 3.2 is reproduced from Table 3.1 with the addition of the highlighting of those solutions in $PfTRUE$ that are not present in $PfAPPROX$. Taking the front shown in Figure 3.10 as an example only the solutions at either end of the front $PfTRUE$ are also present in the front $PfAPPROX$ indicating that for the example provided the value obtained from the error ratio calculation would be 0.4 as only two solutions present in $PfTRUE$ are also present in $PfAPPROX$. The generational non-dominated vector generation (GNVG) metric (Van Veldhuizen, 1997) simply lists the number of solutions in each generation and can be extended to form the non-dominated vector addition metric (NVA) which calculates the difference between two sets $PfAPPROX$ between generations.

<i>PfTRUE</i>		<i>PfAPPROX</i>	
F1	F2	F1	F2
5	1.2	5	1.2
<u>4.5</u>	<u>1.7</u>	4	2.4
<u>3.2</u>	<u>2.1</u>	2.9	2.9
<u>2.7</u>	<u>2.5</u>	1.4	4.1
<u>1.3</u>	<u>4</u>	0.5	6
0.5	6		

Table 3.2 Error Ratio (ER) Table

3.4.2. Metrics Coverage Distance, Coverage and Spread

Practitioners rarely use the ONVGR or ER metric described in the previous section as the sole quality metric. The metrics discussed so far are purely quantitative methods that do not measure the actual Pareto front. For instance Table 3.1 shows that *PfTRUE* has as a member a solution occupying the space at world coordinates $\{1.3, 4\}$. There is no corresponding member with the set *PfAPPROX* but there is however a close neighbour located at world coordinates $\{1.4, 4.1\}$. The following section details a series of heuristics that can be used to attempt to model the quality of solutions.

The error ratio metric may be extended in order to more accurately measure the true relation between the front *PfTRUE* and *PfAPPROX*. The general distance metric (GD) measures the distance between each member of the front *PfAPPROX* and the nearest neighbour (using Euclidian distance) in the front *PfTRUE* with the resultant metric being the average of those distances.

$$GD = \frac{(\sum_{i=1}^C d_i^p)^{1/p}}{|C|}$$

Equation 3.5 Generation Distance Metric

Where C = the number of criteria being considered

And $p = 2$

And d_i = the minimum distance between a member of *PfAPPROX* and a member of *PfTRUE*

Table 3.3 extends Table 3.1 to match solutions present in the front $PfAPPROX$ to the closest neighbour in the front $PfTRUE$. The Euclidian distance is also included. The vector information in the example would provide a generational distance of 0.26.

PfTRUE		PfAPPROX		Euclidian Distance
F2	F1	F2	F1	
5	1.2	5	1.2	0.00
4.5	1.7	4	2.4	0.86
2.7	2.5	2.9	2.9	0.45
1.3	4	1.4	4.1	0.14
0.5	6	0.5	6	0.00
Average Distance (GD)				0.26

Table 3.3 Generational Distance Measures

The maximum Pareto front error (MFE or MPFE) is another metric which attempts to measure the distance between solutions on the fronts $PfTRUE$ and $PfAPPROX$. Where the generational distance metric attempted to measure the distance between the closest solutions on the front $PfTRUE$ the MFE metric uses a reverse logic and presents the largest distance. In the example used elsewhere in this section the MFE error returned would be 0.86 between the solutions located at $\{4.5, 1.7\}$ on PfTRUE and $\{4, 2.4\}$ on PfAPPROX. A value of zero returned would indicate that the two fronts are equal. Deb *et al* (2002) propose the spacing metric which aims to measure the average distance between solutions.

$$S \triangleq \sqrt{\frac{1}{n-1} \sum_{i=1}^n (\bar{d} - d_i)^2}$$

Equation 3.6 Spacing Metric Metric (Ripon, 2007)

Where $n = |PfTRUE|$ or $|PfAPPROX|$ (depending on the set being considered).

And \bar{d} = the mean value of all d_i

And $d_i = \min_j |f_1^i(x) - f_1^j(x)| + |f_2^i(x) - f_2^j(x)|$

Zitzler (1998) introduces the hyper-volume metric as a method of measuring the quality of a given solution space. Beume *et al* (2009) state that the hyper volume metric displays the ability to measure both the proximity of a solution to the front $PfTRUE$ and the spacing along the front $PfAPPROX$. In a solution space where two objectives are considered i.e. the solution space of both $PfTRUE$ and $PfAPPROX$ each member of the solution set can be measured as a rectangle covering the space from $\{0,0\}$ to $\{f(x),f(y)\}$. Where $f(x)$ and $f(y)$ are (assuming a bi-criterion problem) the maximum values seen for each criteria Figure 3.11 and Figure 3.12 present the front $PfTRUE$ previously shown in Figure 3.10 with the addition of the various hyper volume rectangles and complete hyper-volume indicated respectively.

As illustrated in Figure 3.11 and Figure 3.12 a comparative downside to the hyper-volume can be seen in the computational effort required in its calculation. For this reason various authors such While *et al* (2005), While *et al* (2006) and Fonseca *et al* (2006) have sought to develop methodologies for reducing the complexity of the method. Auger *et al* (2012) investigate the use of the weighted hyper-volume metric as a method for generating user preference points. Bader and Zitzler (2008) generate an alternative method where the hyper-volume metric is used as the formation of a fitness function.

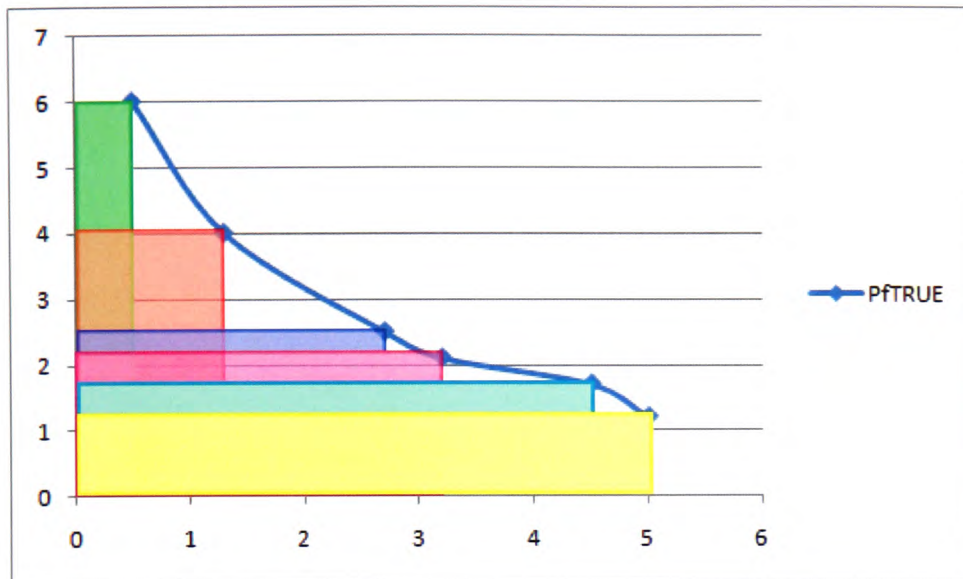


Figure 3.11 Hyper Volume Rectangles

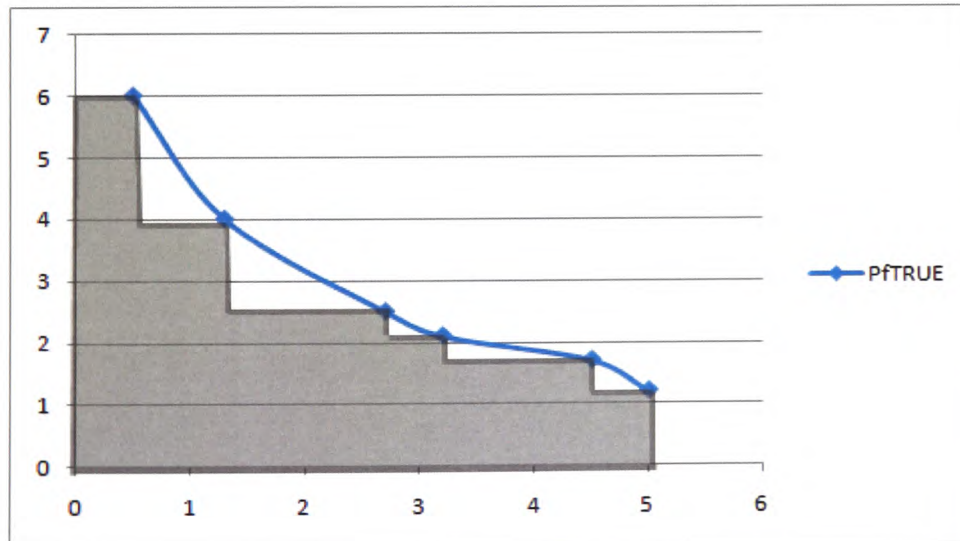


Figure 3.12 Hyper Volume Front

3.5. Chapter Summary

This chapter introduced the various algorithms that have been used to solve multi-objective algorithms and to provide some insight into the complexities involved in measuring the completeness of solutions offered by those methodologies. Evolutionary approaches in the form of Genetic Algorithm dominate the literature regarding multi objective optimisation, though increasingly in recent years ant colony optimisation has become a popular technique. Little research has been undertaken into the use of techniques such as simulated annealing and the tabu search for multi objective optimisation. Where those techniques have been used then the principle methodology has historically been to reduce the multi objective problem into a single criterion using weightings. Only recently have fully Pareto based solutions been sought.

This chapter serves to lay the groundwork for the next chapter in which the primary aim of this work is undertaken, namely the solution of the multi criteria graph optimisation problem using those techniques (EA, the Tabu Search and Simulated Annealing). Reviewing the literature highlights that in many cases where multiple objectives are considered using the Tabu Search and Simulated Annealing, the overwhelming majority of those works use either aspects of the approaches in conjunction with evolutionary approaches (hybrid algorithms), or condense the problem into a single criteria problem using aggregation or weighting. The relatively few works where this is not the case use some form of energy measure or vector to measure the distance from a solution to the approximation of the Pareto front. Despite the fact that several alternative heuristic techniques have been applied to multi objective problems, no evidence of the application of the same techniques having been applied to the problem of the MSPP can be seen. The travelling salesman (Pancero and Mart, 2006), the vehicle routing problem (Banos *et al*, 2012; Zidi *et al*, 2011) and the knapsack problem (Gandibleux and Freville, 2000) are examples of where the various optimisation techniques have been applied to graph based problems. One of the central goals of this research is to attempt to develop these alternative mechanisms addressing specifically the MSPP.

Chapter Four: Experimental Design

4. Experimental Design

This chapter opens with an introduction the datasets against which the various algorithms will be tested. Following on from this brief introduction to the datasets the methodologies employed by the various heuristic algorithms are introduced. Algorithms for the extraction of the optimal front are presented and the mechanism used for path representation introduced. The format of the various datasets used by the algorithms is considered along with any steps involved in the translation from single to multi criteria or from alternative data formats such as geographic road network information. The heuristic methods are then introduced alongside a series of support algorithms. Prior to the introduction of the methodology, data and algorithms used in the study it is useful to reproduce the aims of the project from Chapter 1.

To develop alternative heuristic techniques (to the Genetic Algorithm) for the solution of the MSPP

Assess the ability of those heuristic techniques to solve the MSPP against real world and synthetic graphs

Compare the alternative heuristic approach with algorithmic methods for the solution of the MSPP

Three alternative techniques to the Genetic algorithm approach have been identified. Those techniques are the Simulated Annealing, the Tabu Search and PAES. The final of the three techniques identified, the PAES algorithm, is of interest given its background as the only method implemented for the study that was from conception designed to handle multiple criteria. The Simulated Annealing and Tabu Search techniques are selected due to their widespread use in alternative routing based

applications such as vehicle routing and travelling salesman applications. The Genetic Algorithm has been used elsewhere (as detailed in Chapter 2) for the MSPP and is implemented here in order to provide a comparative base for the selected algorithms.

The requirements of the methodology were developed from an analysis of the existing work detailing the analysis of Genetic Algorithms for the solution of MSPP and as such the methodology employed in this work has been heavily influenced by those same existing works. Recent developments in the application of vector measurement when handling multiple criteria for Simulated Annealing heavily influenced the methodology and implementation of that (Simulated Annealing) approach, together with the Tabu Search. Prior to work detailing the approach vector measurement approaches initial developments for this work consisted of algorithms basic upon the a) selection of a preferred criteria which would act as a tie breaker in cases of indifference or b) combination of the costs into a single value.

4.1. Graph Data Structures

In the current section details of the implementation of the graph structure and algorithms are introduced. C# is an object-oriented programming language that makes it easy and logical to embed functionality regarding each of the “objects” in a graph to individual classes. Figure 4.1 presents a UML schematic of the core graph objects in the developed software. Appendix C includes further UML schematics of the developed system including the domination system, paths and random walking and finally an example of the heuristics in the form of the tabu search. The current section however focuses on the core aspects of graph implementation with a description of how the various objects or classes developed relate to each other and the algorithms developed.

First, we consider the importance of generics to the implemented software. In the simplest definition generic programming is a programming style in which

algorithms are written in terms of to-be-specified-later types that are then instantiated to a given parameter type later as and when needed. The *Graph* and related classes make heavy use of generics. The use of generics in the development of the graph class enables, for all the graph related algorithms, the development the implementation of a single graph object with edge type defined as and when needed by the algorithms. The same graph class can handle edges with no costs, single costs or multiple costs with no amendment to the underlying *Graph* object.

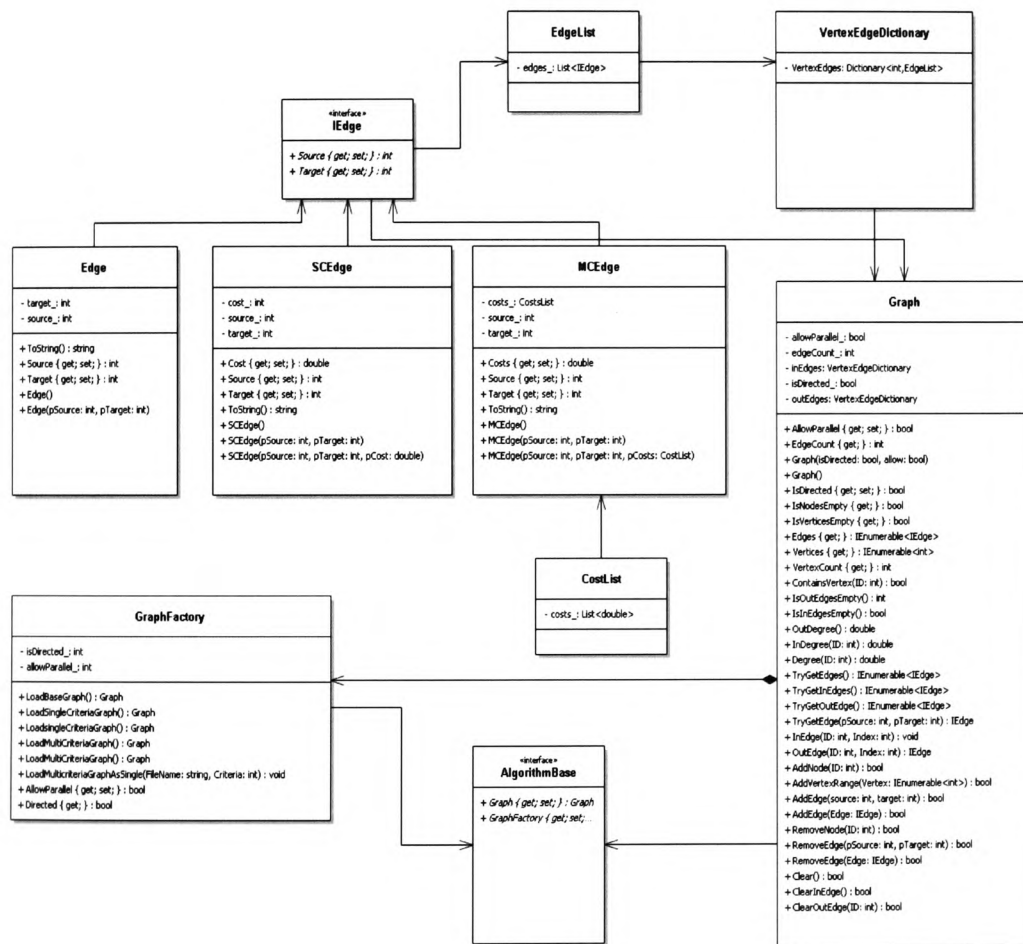


Figure 4.1 UML Diagram Representing the Graph Structure

The *IEdge* interface is, alongside the *Graph* object, a key aspect of the implementation. The interface has member variables for the head and tail of the edge along with accessing properties for both which enable data validation (in the form of basic logic checks to take place). The *Edge* class is the most basic encapsulation of the behaviours of the *IEdge* interface and simply store details of the head and tail, interfaces in C# cannot become variables hence the creation of the separate class. The *SCEdge* class encapsulates the same behaviours but has an additional member variable for the weight or cost of the edge. *SCEdge* is used to represent the edge seen on a single criteria graph. The *MCEdge* also encapsulates the behaviour of the *IEdge* interface but in addition allows the storage of multiple costs through the use of the *CostList* class as a member variable. The *CostList* class inherits the basic behaviours of the *List* collection of the standard C# library. The *EdgeList* class is used to group together one or more edges from the graph. The *EdgeList* class encapsulates the behaviours of the standard *List* Collection in the form *List<IEdge>* therefore the *EdgeList* class can store details of each edge type without modification. The *VertexEdgeDictionary* inherits the basic properties of the C# *Dictionary* class in the form *Dictionary<int,EdgeList>*. Each vertex a graph has an associated entry in the *VertexEdgeDictionary* with the edges being linked to that vertex via the dictionary lookup structure. The *Graph* class stores the details of the incoming and outgoing edges associated with each graph vertex in the *inEdges* and *outEdges* class respectively. The use of the *Dictionary* requires the performance of a validity check prior to edge insertion but allows for fast extraction of incoming and outgoing edges. The *GraphFactory* class enables the reading of graph files from disk as shown in Algorithm 4.1. The factory class itself has two simple boolean member variables that indicate if the graph is directed and if parallel edges are allowed between nodes. The class aims to provide functionality rather than act as a storage mechanism with the entire graph returned from the graph loading mechanisms. The *AlgortihmBase* defines the basic properties of the algorithms used be it single criteria Dijkstra shortest path or multi criteria Simulated Annealing. Each algorithm inherits the basic properties of the *AlgortihmBase* class and therefore has an instance of both the *Graph* and *GraphFactory* classes as properties. The following brief sections of code demonstrate how the use of code generics allow the graph class to handle various edge types without modification.

The remaining classes in the system such as those used to represent paths etc act in the same way, where possible making use of inheritance to reduce code duplication. A *Path* inherits from the *List* (*List<int>*) object with each element of the list representing a vertex visited. *SCPath* (representing a single criteria path) inherits the behaviours of *Path* and with the total cost stored as a member variable. *MCPPath* (representing a multi criteria path) also inherits from *Path* but has a *CostList* as member variable. *Population* inherits from *List* in the form *List<MCPPath>*. The tabu list (*TabuList*) inherits from *Population* but has additional functionality to control the size of the tabu memory. As stated further UML definitions are included in Appendix C but wherever possible software development process makes use of the basic principles of object-orientation. Full source code has been included on Compact Disc with this thesis.

Algorithm:	Examples Of Graph Loading
Input:	NONE
Output:	NONE

```

bool IsDirected = false;

// In actual code this would be a algorithm, not AlgortihmBase as it is an interface
AlgorithmBase AB = new AlgorithmBase();

// Create an instance of a single criteria graph using external GraphFactory
GraphFactory GF = new GraphFactory();
Graph<int,SCEdge> SingleCriteriaGraph = GF.LoadSingleCriteriaGraph("TestSC.gr",IsDirected);
AB.Graph = SingleCriteriaGraph

// Create an instance of a multi criteria graph using external GraphFactory
GraphFactory GF = new GraphFactory();
Graph<int,MCEdge> MultiCriteriaGraph = GF.LoadMultiCriteriaGraph("TestMC.gr",IsDirected);
AB.Graph = MultiCriteriaGraph

// Create an instance of a multi criteria graph (only reading two criteria if more than 2 exist) using internal
//method
AB.GraphFactory.LoadMultipleCriteriaGraph("TestMC.gr",IsDirected,2);

```

Algorithm 4.1 Examples of Graph Loading Procedures

All algorithms are implemented using the Microsoft C# language. The experiments were conducted on a single system. This consisted of an Intel Quad core processor operating at 2.13 GHz and with 4 GB of RAM. The operating system on the machine was Microsoft Windows 7 (64).

4.2. Data Selection

The experimental phase of the project makes use of data sets from two sources. The SPRAND¹ software has been utilized in order to generate a number of random graphs. The edge/vertices ratios selected for those graphs are intended to mimic those that may be found in the real world in the form of roads (Jacob *et al*, 1999). A second set of data formed from real world road networks is also used. The real world road network(s) presented are sections of the UK road network.

4.2.1. Randomly Generated Graphs

The SPRAND software (Cherkassky *et al* 1996) was utilized to generate a series of random graph structures. A number of graph sizes are selected, these range from 100 x 250 (100 vertices and 250 edges) through to 12000 x 36000. The edge/vertices ratios of the randomly generated graph were kept consistent with typical levels found in real world graph structures such as roads. Jacob *et al* (1999) highlight a ratio of around 2.6 in road networks. In Chapter 2 (Section 2.6) the size of a variety of road datasets was considered when comparing road networks with social networks. A typical density ratio of 2.1 – 2.8 can be seen in road networks. In an effort to counteract the presence of any small world clustering five graphs at each size were produced. The SPRAND software was selected for use in this study due to its use in other graph optimization studies where both single and multiple criteria are selected.

¹ SPRAND is the name of the program, and is not an abbreviation.

The SPRAND program generates the underlying graph topology together with a single criterion cost. A further piece of software was developed which produced a user-defined number of additional criteria. The values for these additional criteria were based within a user defined bound range together with the value of the initial criteria generated by SPRAND. The software used to generate additional criteria costs does not manipulate the structure of the graph produced by SPRAND. The method of operation of the software is provided in 4.2.2 specifically Algorithm 4.3. Table 4.1 presents the complete set of graph sizes typically used during the experimental phase of the work. Where necessary for the purposes of a given experiment additional graphs were produced. Where this is the case, the experimental description highlights the fact.

Graph Name	 V 	 E 	Graph Name	 V 	 E
100 X 150	100	150	5000 X 8000	5000	8000
100 X 200	100	200	5000 X 10000	5000	10000
100 X 300	100	300	5000 X 15000	5000	15000
200 X 300	200	300	5000 X 20000	5000	20000
200 X 400	200	400	5000 X 20000	5000	20000
500 X 1000	500	1000	6000 X 9000	6000	9000
500 X 1500	500	1500	6000 X 12000	6000	12000
750 X 1500	750	1500	6000 X 15000	6000	15000
750 X 2000	750	2000	8000 X 9000	8000	9000
1000 X 1500	1000	1500	8000 X 12000	8000	12000
1000 X 2000	1000	2000	8000 X 16000	8000	16000
1000 X 3000	1000	3000	8000 X 20000	8000	20000
2000 X 3000	2000	3000	10000 X 15000	10000	15000
2000 X 4000	2000	4000	10000 X 20000	10000	20000
2000 X 5000	2000	5000	10000 X 25000	10000	25000
3000 X 5000	3000	5000	10000 X 30000	10000	30000
3000 X 6000	3000	6000	12000 X 15000	12000	15000
3000 X 7000	3000	7000	12000 X 24000	12000	24000
5000 X 6000	5000	6000	12000 X 36000	12000	36000
5000 X 7000	5000	7000			

Table 4.1 Random Graph Sizes

4.2.1.1 Calculation of Front *PfTRUE*

On anything but the smallest of graphs enumerating all of the paths between two vertices on a well-connected graph would prove difficult as there is no realistic methodology for quantifying the actual number of paths that exist between those vertices. Mooney (2004) advocates the use of geodesics suggesting that should the vertices exist in the same local neighbourhood then few *optimal* paths will exist. A geodesic path is the shortest path through the network from one vertex to another. Vertex pairings with a high geodesic value will theoretically be outside the local area and more likely to have the property of a large number of optimal paths. Taylor (1999) performs a similar action based upon visual analysis of the network. Algorithm 4.2 from Newman (2001) is used to calculate the geodesic path (shortest number of edges traversed) from a source vertex to all other vertices. The source vertex is stored together with all other connected vertices (destinations) and the geodesic length of that path. The list is stored in reverse order with paths of the highest geodesic value coming first. The process is repeated for the required number of iterations, upon completion of which each of the lists is merged into a single ‘master list’, which is again sorted into descending order of geodesic length. The ‘master list’ forms the basis of vertex pairings which will be subjected to further analysis, in the hope that those paths will consist of fronts with a higher number of solutions being seen as optimal.

A brute force approach is used to acquire the fronts *PfTRUE* on random graphs on vertex pairings with a high geodesic value (taken from the ‘master list’). The K shortest path methodology is performed with a high degree of K. That technique allows for the admission of a large number of paths between two vertices on the graph. In addition the random walk algorithm is allowed to operate over a long period (one hour) in order to produce a substantial number of paths between the two vertices. The Pareto optimal front (*PfTRUE*) is extracted from the combination of both sets of paths. Where $|PfTRUE| > 3$ the source and destination vertices are stored together with the contents (path descriptions) of *PfTRUE*. The process was repeated until 5000 pairs of vertices where $|PfTRUE| > 3$ were discovered, or full graph exploration performed. A value of

$|PfTRUE| > 3$ has been selected for two related reasons. Prior to the introduction of a substantially automatic procedure for the extraction of the set of optimal solutions between two vertices on a graph, a lengthy period of manual extraction on real world graphs took place. For rural roads it became apparent that a large number of vertex pairings selected only have one or two paths present in the set of optimal solutions. It is logical that other road types will also have the same property, that is, only two optimal paths between a randomly selected pair of nodes but the property appears more frequently on rural roads. A value of three was selected to provide a form of filter to the search process. Without the filter ($|PfTRUE| > 3$) any two vertex pairings could be selected. In order to introduce a ‘challenge’ for the heuristics the filter was introduced. In the work we are interested in testing the algorithms ability to seek out or evolve to optimal paths. If there is a limited number, say one or two paths then the challenge set to the algorithms may be illogical. Without the value of the filter put in place performing the analysis would simply be the case of applying any shortest path algorithm to the search process. As the aim of the heuristics employed is to identify high quality solutions yet not specifically the shortest paths in any criteria. Setting the filter value greater higher than the number of criteria increase the likelihood of the pairing providing a challenge to the heuristic algorithms.

4.2.2. Graph Format

The graphs created by the SPRAND software are stored as a series of flat files in a format extended from that used in the various DIMACS (Discrete Mathematics and Theoretical Computer Science) challenges. The output of the SPRAND software presents a single criteria graph with the following properties:

- A graph contains n vertices and m edges
- Vertices are identified by a series of positive integers
- Graphs can independently considered directed or undirected
- Graphs can have parallel arcs and contain self loops

- The costs associated with an edge may be positive or negative, although in the case of the graphs used in this study only positive edge costs are considered with the lower and upper bounds specified during the graph generation process.

Algorithm:	Newmans' Geodesic Distance Calculation Algorithm
Input:	$G = (V , E)$ = A Graph S = The Source Vertex For Geodesic Analysis
Output:	D = Set Of $ V $ Containing Geodesic Distances P = Set Of $ V $ Containing Predecessor Vertices
<pre> STILL_VERTICES_UNVISTED = true WS = 0 FOR (i = 0 TO V) D[i] = ∞ P[i] = i P[S] = S D[S] = 0 WHILE (STILL_VERTICES_UNVISTED) { VERTICES_WITH_DISTANCE = Get Vertices With Distance WS FOREACH (Vertex v In VERTICES_WITH_DISTANCE) { OUTGOING_EDGES = Get Outgoing Edges From v in G(E) FOREACH (Edge e in OUTGOING_EDGE) { IF (D[Head Of e] == ∞) { D[Head Of e] = WS + 1 P[Head Of e] = v } ELSE IF (D[Head Of e] == (WS+1)) P[Head Of e] = v } } WS = WS + 1 STILL_VERTICES_UNVISTED = false FOR (j = 0 TO V) IF (D[j] == ∞) STILL_VERTICES_UNVISTED = true; } </pre>	

Algorithm 4.2 Geodesic Calculation Algorithm (Newman, 2001)

The files output by the SPRAND software consist of five line types. Table 4.2 presents a basic outline of those types. In the work undertaken for this thesis the only line considered are the edge description lines (those starting with ‘a’). The remaining line types are ignored by the developed software. Figure 4.3 presents a simple graph consisting of five vertices and eight edges generated by the SPRAND software while Figure 4.2 presents a visualisation of the same graph definition obtained from the SPRAND software.

Type	Purpose	Example
Comment Line	Comment lines can appear anywhere and are ignored by programs.	c This is a comment
Graph Dimensions	Indicates the number of nodes and edges in the graph	p sp 5 8
Meta Information	Gives background information about the graph	t rd_5_8_2147483647_
Source Node	Indicates the number of arcs in the graph	n 1
Edge Descriptor	Specify an edge, giving the source, target and weight of the edge in that order	a 1 2 3

Table 4.2 SPRAND Attribute Types

Having acquired a basic random graph with single criteria edge costs a further piece of software developed for the study is run in order to generate a user-defined number of criteria. The software itself does not manipulate the graph structure obtained from the SPRAND, and instead merely generates criteria values based upon a user-defined bound set. Figure 4.4 presents the updated graph structure following the application of additional criteria using the software developed and is an example of the input information into the various heuristic algorithms developed for analysis during the experimental phase of this work. The “*GraphFactory*” class described in section 4.1 reads graph information in the DIMACS format ignoring all lines other than edge descriptors. Other than the application of additional comment information, which is ignored by the developed algorithms, and the application of user specified weight information no changes are made to the basic graph structure are made. Algorithm 4.3 highlights the conversion process.

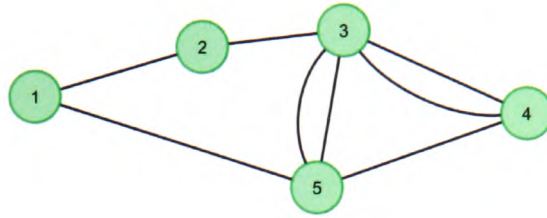


Figure 4.2 Visualisation of Simple SPRAND graph

```

c random network for shortest paths problem
c extended DIMACS format
c
t rd_5_8_2147483647_
c
c
p sp      5      8
c
n      1
c
a      1      2      4316
a      5      1      6093
a      2      3      9317
a      3      4      9126
a      4      5      5229
a      3      4      6283
a      5      3      3126
a      3      5      9568

```

Figure 4.3 Single Criteria Graph from SPRAND

```

c A Multi Criteria Graph
c Converted From "SmallExampleGraph.gr"
c With bounds 1 - 10
a 1 2 1 5 8
a 5 1 3 4 2
a 2 3 5 6 9
a 3 4 4 4 2
a 4 5 2 3 1
a 3 4 7 7 8
a 5 3 3 4 9
a 3 5 2 1 8

```

Figure 4.4 Single Criteria Graph Following Multi Criteria Conversion

Algorithm:	Convert Single To Multi Criteria Graph Algorithm
Input:	<i>PATH</i> = The Name Of The Input Graph Produced By SPRAND <i>OUTPUT</i> = The Name That The Multi Criteria Graph Will Be Saved As <i>MIN</i> = The Lower Bounds Of Each Arc Cost <i>MAX</i> = The Upper Bounds Of Each Arc Cost <i>CRITERIA</i> = Number Of Criteria To Create
Output:	A File On Disk Representing A Multi Criteria Graph
<i>Reader</i> = Open <i>PATH</i> For Reading <i>Writer</i> = Open <i>OUTPUT</i> For Writing <i>R</i> = Random Number Generator Save Comment Information to <i>Writer</i> // A description of the graph – meta data only <i>Line</i> = Read A Line From Reader WHILE (<i>Line</i> != Null) { WHILE (<i>Line</i> Contains “ ”) // Double Spaces Replace Double Spaces In <i>Line</i> With Single Spaces IF (<i>Line</i> Starts With ‘a’) { <i>PARTS</i> { } = <i>Line</i> Split By Spaces // <i>PARTS</i> 0-2 Will be the Arc indicator, head and tail of the Edge. Ignore The Arc Indicator // and existing costs in <i>PARTS</i> [3] <i>NEWLINE</i> = “a ” + <i>PARTS</i> [1] + “ ” + <i>PARTS</i> [2] + “ ” FOR (<i>i</i> = 0 TO <i>CRITERIA</i> -1) <i>NewCriteria</i> = Get A Random Integer Between <i>MIN</i> && (<i>MAX</i> +1) From <i>R</i> <i>NEWLINE</i> = <i>NEWLINE</i> + <i>NewCriteria</i> + “ ” Remove The Last Character From <i>NEWLINE</i> // It will be a space and not needed Use <i>Writer</i> To Save <i>NEWLINE</i> To Disk } <i>Line</i> = Read A Line From Reader } Dispose Of <i>Reader</i> and <i>Writer</i>	

Algorithm 4.3 Conversion to Multiple Criteria Algorithm

4.2.3. Real World Graphs

Road networks for the UK were sourced and reviewed. The selected data forms part of the Integrated Transport Network (ITN) layer of the Ordnance Survey (OS) MasterMap information sets. A series of towns and cities in England and Wales have been selected as base points. Newcastle Emlyn in South West Wales, Cardiff in South East Wales and London in South East England. For each of these three locations a series

of distance intervals ranging from 250 meters through to 6000 meters were selected (taken from the OS Centroid coordinate for that location) and the road network and road restriction information extracted for that distance interval. For certain experiments a given distance interval value may not be considered for a location. In the following section the reason for selecting a particular location is presented together with a description of the data. The scale of the graphs used is 1:1250 for urban areas and 1:2500 for rural areas. Other datasets of road network information have been considered for analysis. Alternatives reviewed consisted of the OS Strategi and Meriden products consisting of 1:250,000 and 1:50,000 data sets respectively. Those two products however lack the ‘completeness’ of the Mastermap ITN layers failing to include features such as alley ways and private roads. Table 4.3 provides details of the road types available via the OS ITN network layer.

International data sets can be found in the forms of US Tiger/LINE data, a set freely available for each of the 50 US States or the US National Highway planning networks, again freely available for each of the 50 US States at a scale of 1:100,000. Zhan and Noon (2000) make use of multiple resolutions of various US state road networks. The information sets used by Zhan and Noon are no longer available. The OpenStreetMap project allows for the export of crowd sourced (Brabham, 2008) data for various location around the world and for the extraction of sub sets of that information. Following a review of the available datasets, the stated areas were selected for analysis. A number of reasons provided a rational for this decision, largely notably the completeness of the datasets themselves and coverage of geography types.

The extraction of datasets from Mastermap tiles resulted in a regular shaped datasets in the form of a square of a set distance around the Centroid of the selected area. The regular shape of the data tiles retrieved may be considered unusual. The alternative would have been to consider human based boundary information such as those seen in local authority or census area outlines. Ease of use was the primary deciding factor when selecting the regular grid of Mastermap together with a greater

degree of control of the geographic area considered. The choice either of either approach would not have any effect on the performance of the algorithms.

Road Type
Motorway
A road – dual carriageway
A road – single carriageway
B road
Minor road
Local street
Alley
Private road – publicly accessible
Private road – restricted access
Pedestrian Street

Table 4.3 Road Types Available via the Mastermap ITN Layer

4.2.3.1 Newcastle Emlyn

Newcastle Emlyn (Castell Newydd Emlyn) is a town bordering the counties of Ceredigion and Carmarthenshire in West Wales and lying on the River Teifi. Data for the area has been selected due to the rural nature of the town and the area surrounding it. Table 4.4 gives the graph sizes selected. Figure 4.5 provide a high-level overview of the covered area. The information presented in Figure 4.5 is at a higher scale than the ITN layer and therefore not all structures are visualised in that figure.

OS Centroid Position (OSGB36): 230500, 240500

Graph Name	Distance (KM SQ)	Features	Edges	Vertices	Density
NE 1000	1000 (4)	146	2722	1352	2.01
NE 2000	2000(8)	219	5420	2692	2.02
NE 4000	4000(16)	618	16314	8098	2.01

Table 4.4 Newcastle Emlyn Statistics

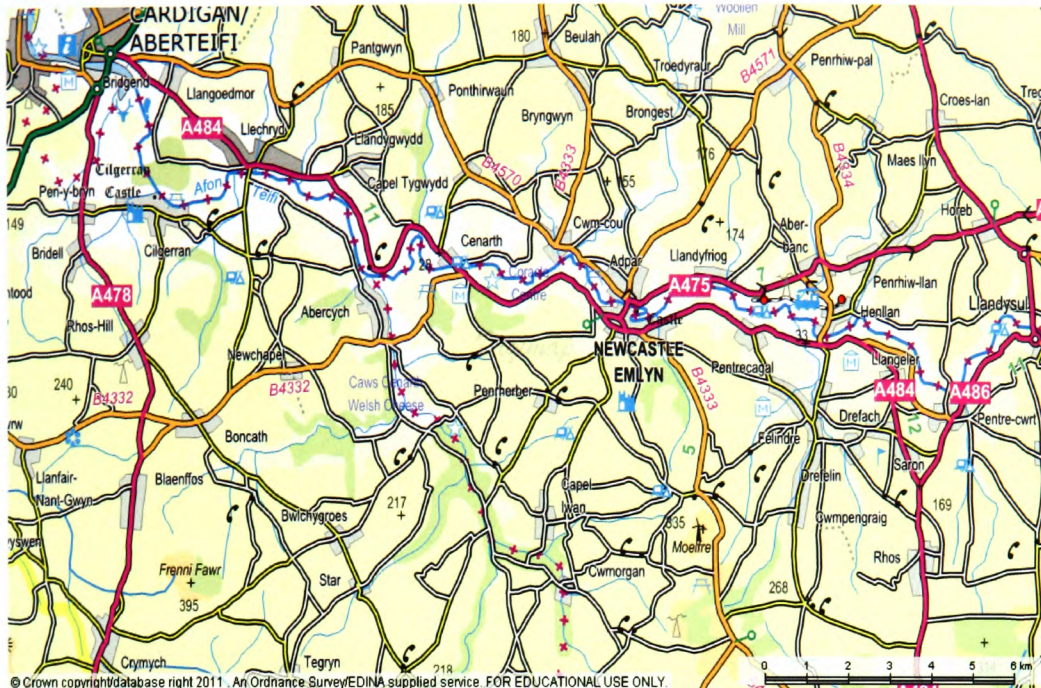


Figure 4.5 General Overview of Newcastle Emlyn (Crown Copyright, 2011)

4.2.3.2 Cardiff

Cardiff (Caerydd) is the capital and largest city of Wales and is located on the South Wales coast. The city is surrounded by a number of rural areas separated from urban areas by geographic features such as mountains and rivers. Large roads (the M4, the A470 etc) act as connectivity corridors linking smaller pockets of urban environment. The sea forms a natural barrier for much of the city. The area has been selected due to the rural/urban mix in conjunction with the connectivity issues highlighted. Table 4.5 gives the graph sizes selected.

OS Centroid Position (OSGB36): 318500, 176500

Graph Name	Distance (KM SQ)	Features	Edges	Vertices	Density
Cardiff 250	250 (1)	75	560	275	2.03
Cardiff 500	500 (2)	318	2288	1076	2.12
Cardiff 750	750 (3)	666	4526	2106	2.14
Cardiff 1000	1000 (4)	1171	7278	3366	2.16
Cardiff 1500	1500 (6)	2555	14428	6603	2.18
Cardiff 2000	2000 (8)	4266	23762	10798	2.2
Cardiff 4000	4000 (16)	9486	62486	29044	2.15
Cardiff 6000	6000 (24)	16327	123428	58383	2.11

Table 4.5 Cardiff Statistics

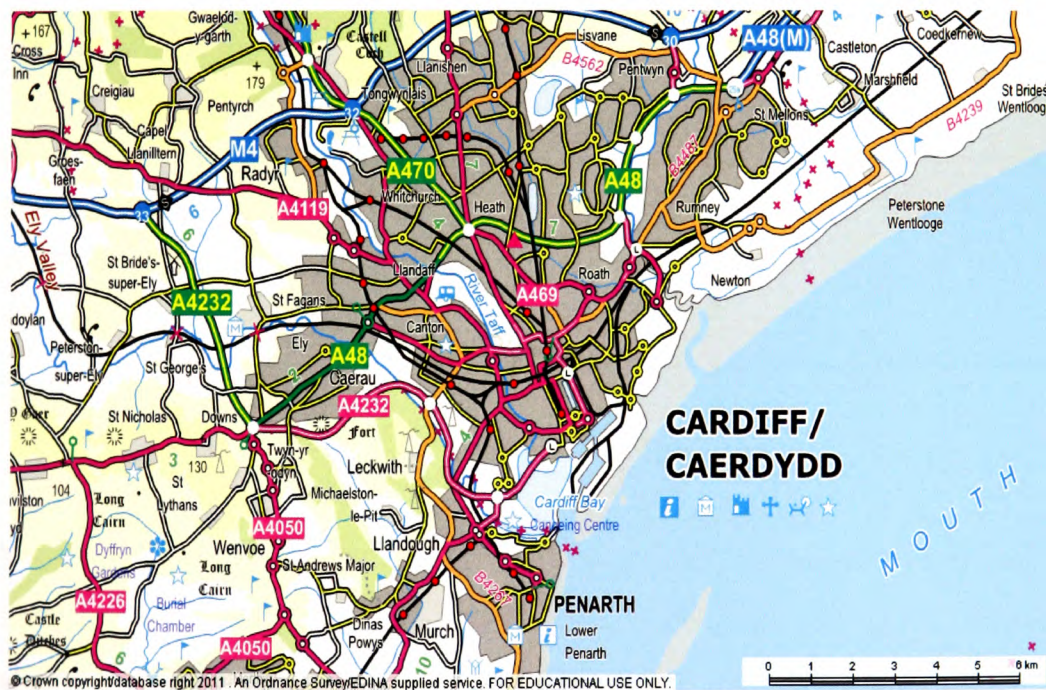


Figure 4.6 General Overview of Cardiff Area (Crown Copyright, 2011)

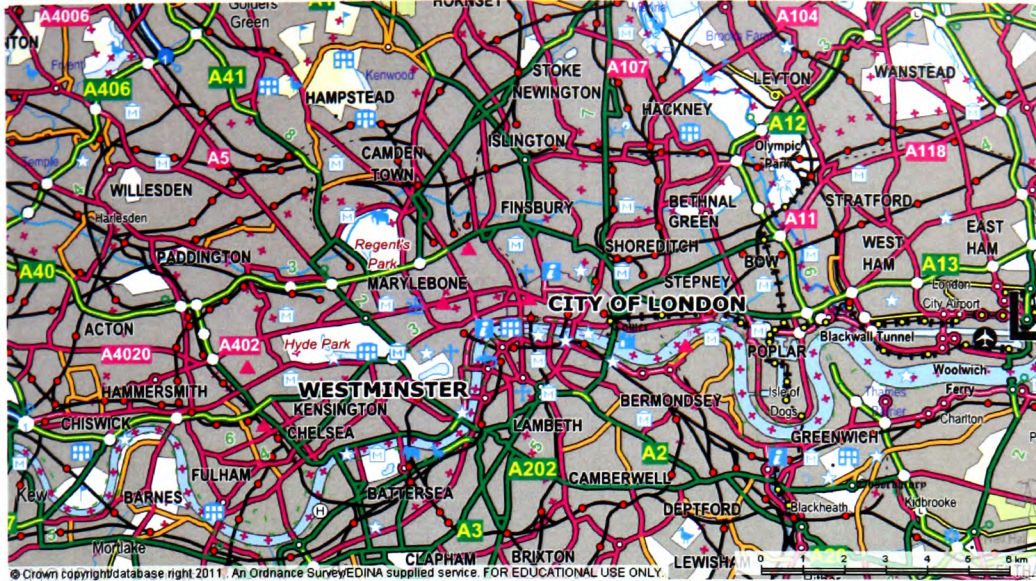


Figure 4.7 General Overview of London Area (Crown Copyright, 2011)

4.2.3.4 Real World Data Observations

The aim of Figure 4.8 and Figure 4.9 is to highlight the comparative edge and vertex density between each of the three areas under consideration. The London based data sets exhibit a much higher level of density than the other locations. As is to be expected given its rural nature the Newcastle Emlyn data exhibits a much lower density of both vertices and edges.

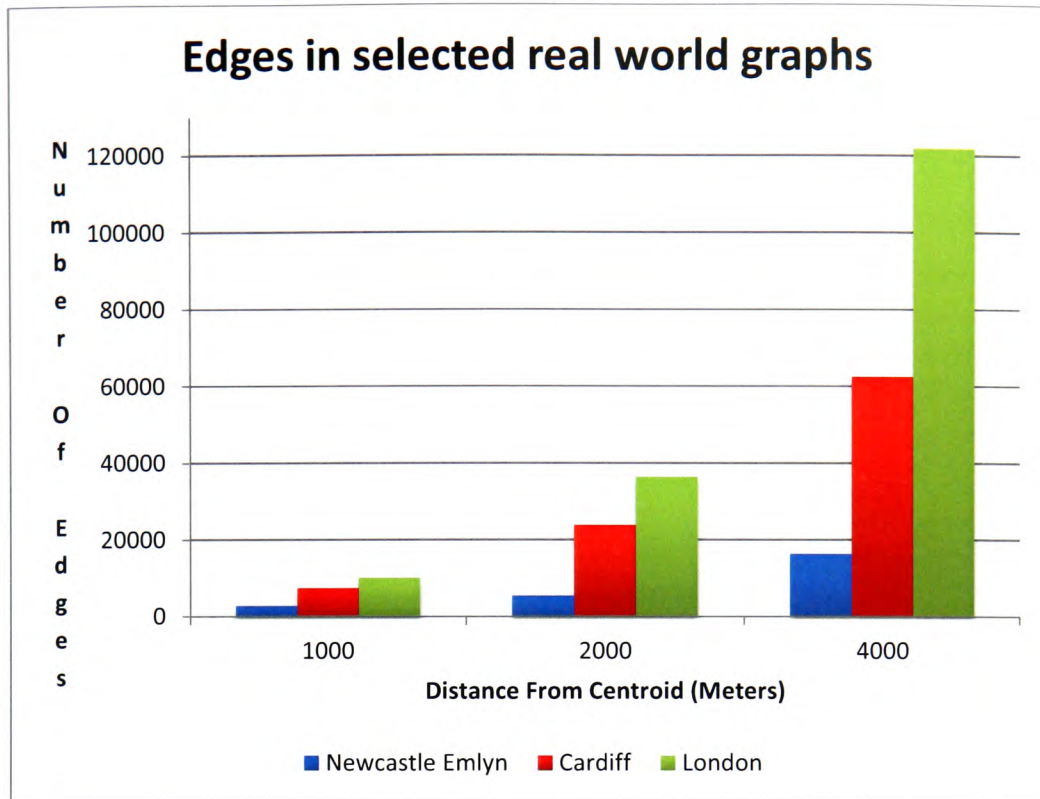


Figure 4.8 Edges in Selected Real World Graphs

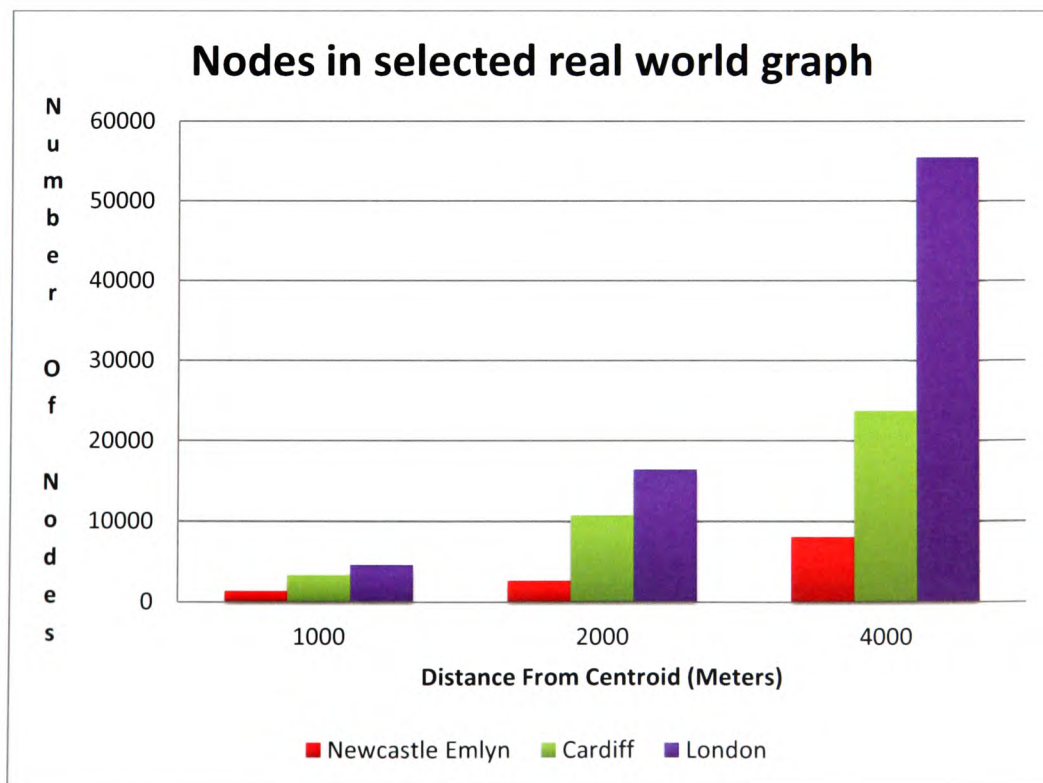


Figure 4.9 Vertices in Selected Real World Graphs

4.2.3.5 Calculation of Front *PfTRUE*

When calculating the front *PfTRUE* on real world graphs a subtly different approach is taken as when compared to when calculating *PfTRUE* on random graphs. On real world graphs a high geodesic value will not always correspond to the high admission of optimal paths into *PfTRUE* due to factors such as the sinuosity of the road (Balboa and Lopez, 2008 and Bagheri *et al*, 2005).

The nature of the translation of the road network to a digital model involves some process of digitization. Depending on the location and structure of the road the number of points making up a road link can vary. Slight turns in a road may result in a greater number of vertices representing the two lengths of road measuring the same straight-line distance. Hence, two road links of the same straight line distance may have substantially different geodesic levels. The principle is visualized in Figure 4.10 and Figure 4.11. The figures show how despite the direct distance between two vertices being the same the road topology can vary greatly resulting in a high degree of difference in the geodesic values. Further to that point Table 4.4, Table 4.5 and Table 4.6 introduced the notion of road ‘Features’. A road may be split up in several sections or road links known as ‘Features’. Individual road links are generated when one (or more) of the following conditions (OS 2012) are met:

- The intersection or crossing of carriageways
- The location where a road name or number changes
- The location where a road name or number ceases to apply
- The start or end of a carriageway
- If a section of a road between junctions is subject to a ‘one-way’ restriction, that section will be given a start and end vertex and becomes in effect a new link.

OS (2012)

In addition to the above rules regarding road geometry and feature makeup each link appears to end at some arbitrary value in terms of length. In Table 4.7 the degree to which the number of vertices varies between road link features in rural and urban environments is highlighted. The aim of the table is to demonstrate how the geodesic path length is not always a realistic model of an actual complexity when applied to real world graphs. Following Table 4.7 an alternative method is suggested for use on real world graphs.

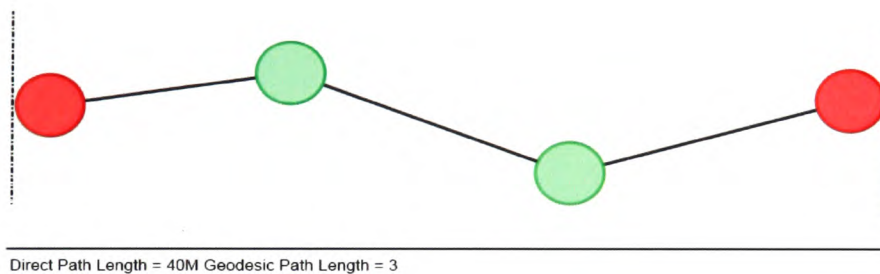


Figure 4.10 Almost Straight-Line Path Requiring Few Links

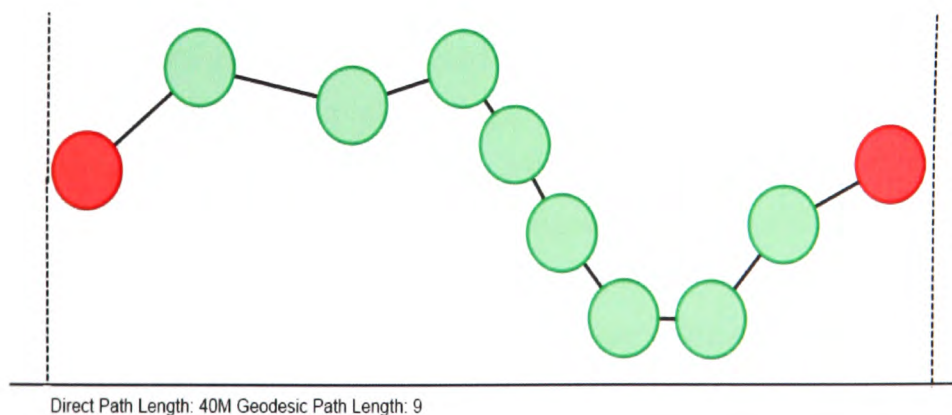


Figure 4.11 Road Feature with High Geodesic Distance

Graph	Features	Vertices	Vertices Per Feature
NE1000	146	1352	0.11
NE2000	219	2692	0.08
NE4000	618	8098	0.76
London1000	1577	4610	0.34
London2000	6170	16474	0.37
London4000	20847	55496	0.37

Table 4.7 Feature and Vertex Ratios on Urban and Rural Roads

An alternative method of calculating *PfTRUE* has been developed. The aim of the new methodology is the same as that used on random graphs i.e. to identify vertex pairings with a higher number of optimal paths between those two vertices. The methodology developed makes use of the Dijkstra's shortest path algorithm. In the tests carried out for this work a high performance open source implementation based upon the Boost (2012) graph library has been used. The algorithm calculates the shortest path for each criterion (distance, travel time and road link type). In those cases where three paths for each of those criteria are not equal the vertex pairing is exported to disk together with each of the calculated paths. The procedure is performed until a user-defined number of edge-vertex pairings are considered or the entire graph has been explored. In this work that limit is set to fifteen million. For many of the smaller graphs full exploration was performed. It should be noted that where the shortest paths are not equal the front *PfTRUE* will always consist of at least n solutions where n is the number of criteria under consideration where the shortest paths differ.

Having initially processed the graph using the Dijkstra shortest algorithm there is now a data source containing each pairing along with the shortest path for each criterion. The follow-up process is to calculate the path cost for each of the remaining criterion travelling the same path. For instance, where the path has been optimised for distance the costs of travelling the same path in terms of the additional criteria (travel time and road type) are generated and again stored. The absolute delta values (ADV) for each path are then generated and the list of vertex pairings sorted from highest to lowest in terms of those delta values. The procedure operates on the principle that where the delta value is larger there is a greater potential to admit a wider spread of Pareto optimal solutions. The absolute delta value is defined in Equation 4.1.

$$ADV = \forall c \in M: f(a)^c - f(b)^c$$

Equation 4.1 Absolute Delta Calculation

Where c is the current criteria and M is the total number of criteria. $f(a)$ and $f(b)$ are the costs of travelling a path using criteria c .

The process then moves to employ a method similar to that seen in the Climaco and Martins algorithmic approach to solving the MSPP. K shortest paths are generated until the value of a path on a selected criterion is greater than that of the shortest path on each of the remaining criteria. However, unlike the Climaco and Martins approach the process is repeated for each criterion. Each path generated by the K Shortest path algorithm is admitted to an external archive. When the process of path generation is completed the front $PfTRUE$ can be extracted. The method allows the selection of a minimum number of solutions in the set $PfTRUE$. In this work only vertex pairings where a minimum of three members are present in $PfTRUE$ are stored along with the paths and values of $PfTRUE$ for future analysis during the experimental phase. This section has outlined a process of $PfTRUE$ generation that operates in such a way as to ensure data is gathered within a comprehensive and robust framework. The process is

continued until 5000 vertex pairings where $|PfTRUE| > 3$ have been discovered or full graph exploration performed. The algorithm used to identify the exact set of optimal solutions between two nodes with three criteria is provided in Algorithm 4.4. A rationale for $|PfTRUE| > 3$ has been provided in section 4.2.1.1.

4.2.3.6 Data Extraction Process

The road information datasets were gathered from the DIGIMAP on-line service. Digimap is a web mapping and online data delivery service developed by the EDINA national data centre for UK academia. It offers a range of on-line mapping and data download facilities which provide maps and spatial data primarily from the OS but also includes other sources such as the British Geological service. The DIGIMAP service supplies the OS Mastermap datasets selected for this work in the Geographic Mark-up Language (GML) format. GML is a common standard format used for the exchange and transfer of spatial information sets that whilst being detailed is also difficult to interpret. In the interests of efficiency an existing piece of software, InterpOSe, was used to convert from the provided GML format into ESRI Shapefiles. The InterpOSe software is recommended by the OS for and conversion of simple data sets such as those used in this study. Each dataset was separately processed and stored. Having acquired the datasets and converted them into ESRI Shapefiles a separate piece of software was developed in order to translate the datasets into the extended DIMACS graphs. The developed software makes use of a number of open source libraries, notably the “SharpMap”² mapping library and the “NetTopologySuite” (NTS) spatial analysis library. The SharpMap library is used to visualise the road network with NTS used to assist in the translation of the network information into a graph.

² Declaration: The author of this work is a developer on the SharpMap library

Algorithm:	Extraction Of PfTRUE From Real Roads Using Three Criteria
Input:	S = The Identifier of the source vertex D = The Identifier of the target vertex $G = (V , E)$ = The Graph To Be Analysed $FILTER$ = Minimum number of solutions required in PfTRUE // Default = 3
Output:	$PfTRUE$ = Exact Set Of Optimal Solutions Between S And D $POTENTAILS = \{ \}$ // Set Of Paths Between S and D
P^D = Calculate the shortest path between S And D using Dijkstra algorithm using distance cost $POTENTAILS = POTENTAILS + P^D$	
C^D = Calculate the cost of traversing P^D using the distance metric in G C^T = Calculate the cost Of traversing P^D using the travel time metric in G C^R = Calculate the cost of traversing P^D using the road type metric in G K^T = Calculate the shortest path between S and D using K shortest path algorithm using travel time cost WHILE $K^T \leq C^T$	
{ $K^T = K^{T+1}$ // Calculate the next shortest path using travel time cost IF (! $POTENTAILS.Contains(K^T)$) $POTENTAILS = POTENTAILS + K^T$ }	
K^R = Calculate the shortest path between S and D using K shortest path algorithm using road type cost WHILE $K^R \leq C^R$	
{ $K^R = K^{R+1}$ // Calculate the next shortest path using travel time cost IF (! $POTENTAILS.Contains(K^R)$) $POTENTAILS = POTENTAILS + K^R$ }	
$PREFILTERED$ = Extract Optimal Solutions From $POTENTAILS$	
IF $ PREFILTERED > FILTER$	
Return $PREFILTERED = PfTRUE$	
ELSE	
Return $NULL$	

Algorithm 4.4 Extraction of PfTRUE from Real Road Graphs

Figure 4.12 presents an overview of the software with one of the graphs used in this study (Cardiff250) displayed in the developed software. With minor modifications the system could be further enhanced to provide Google™, Bing™ or OpenStreetMap information as a backdrop to the road network. However the modifications would require the re-projection of the OSGB36 road networks into the format used by those online mapping service. In the interest of efficiency that activity was not undertaken as the modification would only serve as a visual aid.

The software consists of three functions; the first of the three simply allows the Shapefile information to be accessed by the system. The second allows for the variation of various road network parameters. The third and final function translates the road network into the graph structures into a format compatible with the “*GraphFactory*” functionality described previously in this chapter (4.1). Figure 4.13 presents a view of the various graph parameters that may be controlled during the translation process. The system allows for the automatic removal of road types in addition to manually setting a travel speed overriding the built in defaults. A final option allows each road type to be given a preference value; setting motorways to high value would reduce the likelihood of optimal paths being seen to traverse road links matching that type; setting the preference rate to a lower value in comparison to other road types would increase the likelihood of optimal paths traversing motorways.

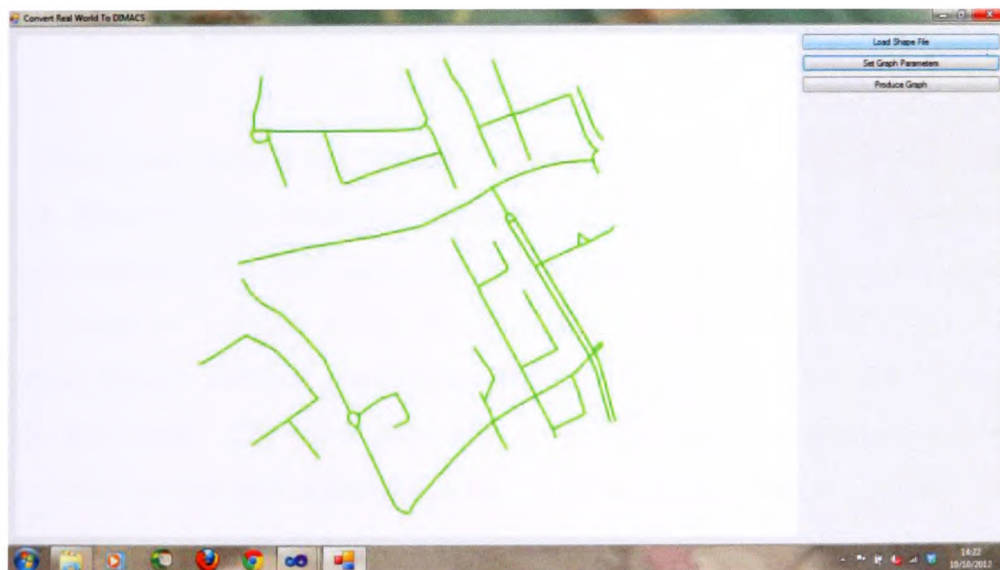


Figure 4.12 Overview of Translation Software

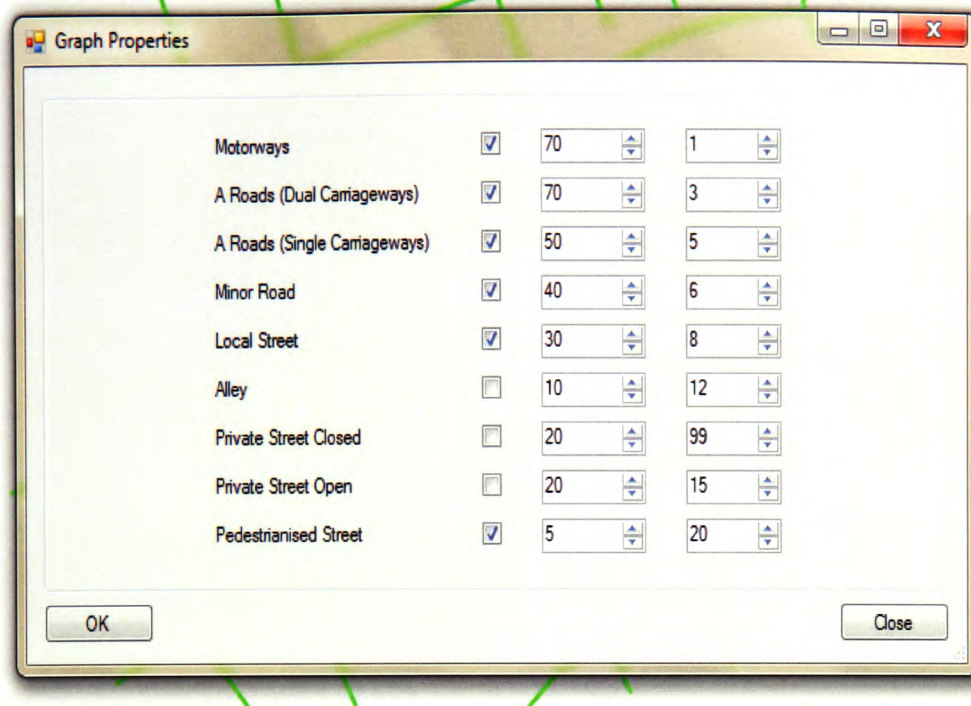


Figure 4.13 Parameter Setting for Translation Software

During the analysis on real road networks two criteria are the primary focus of attention, those being the distance of the path along the road network together with the time spent travelling the same path. The introduction of the preference attribute allows for the extension to three criteria as previously described. The ITN layer of the Mastermap datasets includes information related to the length of the road link together with the road type. The travel time cost is derived from the length of a road link together with its road type and speeds assigned for travelling a road of that type. During the data selection process data sets with accurate speed limit information was initially sought, however no such data sets could be identified. The OpenStreetMap road networks allow maximum speed limits to be included as attribute based information but such information is incomplete. The TIGER datasets previously described identify a travel time associated with each edge of the graph however the methodology used to produce the travel time value is the same as that used here i.e. the travel time value is a function of the road type and length of the edge.

Figure 4.13 presents an example set of parameters. In the provided example the translation software would include all road types in the graph with the exception of those identified as alleyways and private streets. The translation would include pedestrian streets as being traversable although very slowly (5 MPH). Motorways and dual carriageways would be traversable at 70MPH. Local streets are traversable at 30MPH. If three criteria were considered then the optimisation algorithms would theoretically limit the use of pedestrianised streets due to comparatively low preference value associated with it.

In order to translate the information contained in the Shapefile into a suitable graph structure a multi-pass procedure is used. The system first scans each road object in turn; storing the geographic coordinates in a lookup table with each assigned a unique identifier. Have stored each coordinate in the lookup table a second pass is made. During the second pass the methodology again scans each road feature. For each geographic node in the feature, it retrieves the unique identifier associated with a geographic coordinate along with that of the following coordinate in the road feature and creates an edge between those two identifiers. The software identifies the distance between the two nodes and associates that information with the edge details. It then calculates the travel time between the using the values retrieved from the user preferences as shown in Figure 4.13. A simple lookup table enables the preferences value for each road type to be retrieved. The algorithm then saves the complete edge (head, tail, distance, travel time and preference value) to disk using the DIMACS format. Algorithm 4.5 presents the methodology in a pseudo code form. In a final step, the algorithm saves the geographic information into a separate file on disk. Figure 4.14 gives an example of the format used to when saving the coordinate information. A small example of the Cardiff 250 graph used in the study is provided. The information is saved in the format used by the ninth DIMACS challenge. Coordinate lines are of the form 'c *X Y id*' where *X*, *Y* and *id* are the x-coordinate, the y-coordinate and the identifier of the node respectively. *c* is used to identify a coordinate descriptor. A small subset of the Cardiff 250 graph information is given in Figure 4.15. The translation software will, as currently implemented, only convert MasterMap information sets

translated from the original GML format to ESRI Shapefile using the InterpOSe software. The complete Cardiff250 graph and coordinate information is provided in Appendix D of this thesis as an example.

In addition to requiring the graphs to be converted from GML to Shapefile using the InterpOSe software the algorithm developed to convert the Shapefile to DIMACS format does not maintain vertex identifiers across graph sizes. In effect what constitutes a coordinate with the identifier of 1 in one translated graph files will not with any degree of certainty have the same identifier in a different graph file. This factor is ignored in the study as each graph is considered independent of another. It would be possible to maintain such node ordering by considering the graphs of each location in reverse ordering of size, processing the largest of the graphs first and terminating processing after the final and smallest of the Shapefiles has been processed. If further factors were of importance such as considering disk storage requirements or the requirement to store the graphs in an external database then the proposed additional processing step may be considered of benefit. The largest of the graphs converted for use in this study (Cardiff 6000) requires 3 megabytes of disk storage space. The conversion procedure performs quickly completing in 0.34 seconds for the largest of the graphs - Cardiff 6000. Therefore the additional processing step does not appear to offer any practical value.

```

c 318602 176273 1
c 318585 176259 2
c 318565.909 176240.775 3
c 318555.029 176230 4
c 318552 176227 5
c 318515 176186 6
c 318500 176169 7
c 318487 176154 8
c 318478.306 176138.07 9
c 318403 176266 10
c 318420.053 176230 11
c 318457 176152 12
c 318462 176146 13
c 318467 176142 14
c 318473.049 176139.901 15

```

Figure 4.14 Coordinate Information for Cardiff250 Subset

```

c Graph Converted From Cardiff250.shp
a 1 2 22.02 1.652 9
a 2 3 26.39 1.98 9
a 3 4 15.31 1.148 9
a 4 5 4.26 0.32 9
a 5 6 55.23 4.142 9
a 6 7 22.67 1.7 9
a 7 8 19.85 1.489 9
a 8 9 18.15 1.361 9
a 10 11 39.83 4.481 10
a 11 12 86.31 9.71 10
a 12 13 7.81 0.879 10
a 13 14 6.4 0.72 10
a 14 15 6.4 0.72 10
a 15 9 5.57 0.626 10

```

Figure 4.15 Graph Information for Cardiff250 Subset

Algorithm:	Convert Road Network To DIMACS Graph Algorithm
Input:	<i>FILENAME</i> = The Name Of The New Graph <i>FEATURES</i> = A Set Of Geographic and Attribute Information Details Sections Of Road <i>PARAMETERS</i> = The User Specified (May Be Default) Parameters For Graph
Output:	A File On Disk Representing A Multi Criteria Graph

GraphWriter = Open *OUTPUT* For Writing
CoordinateFileName = Amend *OUTPUT* To Indicate Coordinate Information // Extension becomes .co
CoordinateWriter = Open *CoordinateFileName* For Writing
Save Comment Information to *GraphWriter* // A description of the graph – meta data only
Save Comment Information to *CoordinateWriter*

COORDINATES = { }
FEATURE_COUNT = |*FEATURES*|
CURRENT_NODE_ID = 1

FOR (*i* = 0 TO *FEATURE_COUNT*)
{
 NODE_COUNT = |*FEATURES*[*i*]|
 FEATURE = *FEATURE*[*i*]
 FOR (*j* = 0 TO *NODE_COUNT*)
 {
 IF (*COORDINATES* Contains *FEATURE*[*j*] == false)
 {
 COORDINATES = *COORDINATES* + *FEATURE*[*j*] + *CURRENT_NODE_ID*
 CURRENT_NODE_ID = *CURRENT_NODE_ID* + 1
 }
 }
}

FOR (*i* = 0 TO *FEATURE_COUNT*)
{
 NODE_COUNT = |*FEATURES*[*i*]|
 FEATURE = *FEATURE*[*i*]
 SPEED = Gather Speed Information From *PARAMTERS* For *FEATURE*[*i*]
 PRIORITY = Gather Priority Information From *PARAMTERS* For *FEATURE*[*i*]
 FOR (*j* = 0 TO *NODE_COUNT*-1)
 {
 EDGE_DISTANCE = Calculate Edge Distance Between *FEATURE*[*j*] + *FEATURE*[*j*+1]
 EDGE_TIME = ((*EDGE_DISTANCE* / 1.609344) / *SPEED*)
 EDGE = "a " + *COORDINATES*[*FEATURE*[*j*]] + " " + *COORDINATES*[*FEATURE*[*j*+1]]
 + " " + *EDGE_DISTANCE* + " "
 + *EDGE_TIME* + " " + *PRIORITY*
 Save *EDGE* To Disk Using *GraphWriter*
 }
}

FOREACH (*COORDINATE* IN *COORDINATES*)
{
 TEMP = "v " + *COORDINATE.ID* + " " + *COORDINATE.X* + " " + *COORDINATE.Y*
 Save *TEMP* To Disk Using *CoordinateWriter*
}

Dispose Of *GraphWriter* And *CoorindateWriter*

Algorithm 4.5 Real Road Extraction Algorithm

4.3. Path representation

In the algorithms considered for this thesis a path between two vertices in a graph is represented as a series of positive integers that indicate the identification number of the vertices through which a route will travel. The first vertex of the path is reserved for the source vertex; the last vertex is reserved for the destination vertex. The number of vertices making up a path is dynamic. Figure 4.16 presents a small graph with a path between two vertices (vertices 1 and 8) highlighted. Figure 4.17 presents the resulting path vector between the source and destination vertices.

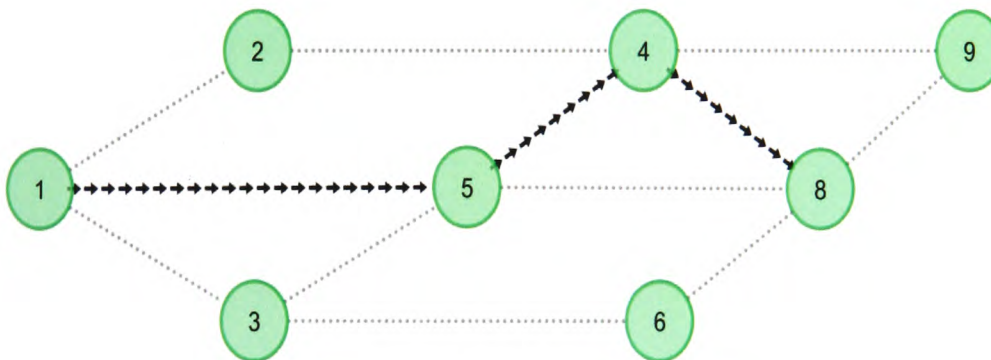


Figure 4.16 Simple Example Graph

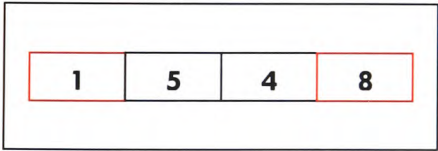


Figure 4.17 Highlighted Path From Figure 4.16

In the work of Larranga *et al* (1999) several methods of representing paths are reviewed. These include the binary structure where paths are represented by the bit patterns making up the ID value (Figure 4.18 presents the binary string view of the path shown in Figure 4.17), the simplicity of the tour based presentation is ideally suited to the MSPP problem (Larranga *et al*, 1999). The representation is computationally efficient with few computational resources being required for any encoding or decoding process. Larranga argues that the tour or path based representation is favourable stating that the most natural representation of one tour is dominated by path representation, and that fundamental reasons lie in its intuitive representation as well as the good results obtained by using it.

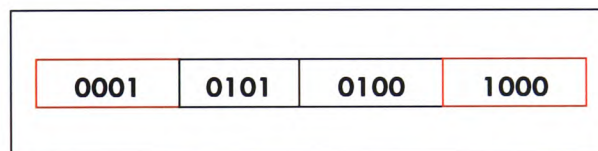


Figure 4.18 Binary String Path Representation

4.4. Random Path Generation

Each of the heuristic techniques used in this work requires the generation of paths between the source and destination vertices. Algorithm 4.6 provides an outline of the random walk used in the study to generate a random path between vertices s and t .

In the approach to the random walk taken in this work a walk starts at vertex s . A vertex is selected at random from the possible neighbours and added to the path description vector, the selected vertex is made current, its neighbour vertices examined and another randomly selected. The process is continued until the destination (vertex t) has been found to be a neighbour of a vertex at which point the target or destination vertex is added to the path description vector (PDV).

Algorithm:	Random Walk Algorithm
Input:	$G = (V , E)$ = A Graph S = The Source Vertex For The Random Walk T = The Target Vertex For The Random Walk $ATTEMPT_SELF_AVOIDENCE$ = Boolean Operator Indicating If The Path Should Attempt to avoid itself where possible
Output:	P = A Random Walk Path Between S And T On Graph G
<hr/> $P = \{ \}$ $P = P + S$ $CURRENT_VERTEX = S$ OUT_EDGES = Get Edges With Tail S $NEIGHBOURS$ = Head Vertices Of Edges In OUT_EDGES WHILE (T Not In $NEIGHBOURS$) { IF ($ATTEMPT_SELF_AVOIDENCE$) { IF ($ NEIGHBOURS > 1$) { $PREVIOUS_VERTEX = P-1$ Remove $PREVIOUS_VERTEX$ From $NEIGHBOURS$ $CURRENT_VERTEX$ = Randomly Select A Vertex From $NEIGHBOURS$ $P = P + CURRENT_VERTEX$ OUT_EDGES = Get Edges With Tail $CURRENT_VERTEX$ $NEIGHBOURS$ = Head Vertices Of Edges In OUT_EDGES } } ELSE { $CURRENT_VERTEX$ = Randomly Select A Vertex From $NEIGHBOURS$ $P = P + CURRENT_VERTEX$ OUT_EDGES = Get Edges With Tail $CURRENT_VERTEX$ $NEIGHBOURS$ = Head Vertices Of Edges In OUT_EDGES } } $P = P + T$ P = Perform Remove Cycles Algorithm On P <hr/>	

Algorithm 4.6 Random Walk Algorithm

Following the discovery of the target vertex the PDV is examined for loops and cycles. If any loops are found they are removed and the final path returned from the random walk method. Algorithm 4.7 outlines the methodology for removing cycles and loops from the candidate path. The approach is classified as a ‘self repairing’ random walk where the walk is repaired after the target vertex is scanned. An alternative can be seen in the ‘self avoiding’ random walk where the algorithm make a ‘conscious’ effort to avoid the creation of loops. The aim of both is to produce a walk between vertices on a graph with no cycles or loops present.

Algorithm:	Remove Cycles Algorithm
Input:	P = Path To Remove Cycles From
Output:	P' = P With Cycles Removed
<hr/> $P' = P$ $COUNT$ = Number Of Vertices In P' HAS_LOOPS = true WHILE (HAS_LOOPS) { $COUNT$ = Number Of Vertices In P' HAS_LOOPS = false FOR ($i = 0$ TO $COUNT$) { Vertex $v = P'[i]$ j = First Position Of v in P' k = Last Position Of v in P' IF ($j \neq k$) { SUB_PATH_J = Copy From $P'[0]$ To $P'[j]$ SUB_PATH_K = Copy From $P'[k]$ To $P'[\text{Length Of } P']$ $REPLACEMENT = SUB_PATH_J + SUB_PATH_K$ $HAS_LOOPS = \text{true}$ $P' = REPLACEMENT$ Break // Exit From FOR Loop } } } <hr/>	

Algorithm 4.7 Remove Cycles Algorithm

In the experiments on the Genetic Algorithm an alternative path generation is tested in the form of the K geodesic where a K series of paths with increasing geodesic values are selected. The algorithm effectively sets the edge cost to one and aims to sacrifice the randomness of the walking technique whilst gaining computational efficiency.

4.5. A Genetic Algorithm for the MSPP

The approach taken to the Genetic Algorithm is heavily influenced by the work of Mooney (2004). The major variation can be seen in the approach taken with the random walk methodology. Mooney makes use of a random walk technique that can best be described as a ‘self avoiding’ methodology. In the current work use is made of a ‘self repairing’ random walk. It is suggested that the ‘self-pairing’ mechanism will offer a performance advantage over the ‘self avoiding’ method which will be particularly evident on real world graph data which will often include ‘dead ends’ which the ‘self-avoiding’ methodology will find difficult to cope with. The concept is illustrated through the use of Figure 4.19. Either random walk methodology may encounter vertices that would introduce dead ends (given in blue). The self-avoiding random walk would if any such vertices were encountered require the regeneration of the entire path. The ‘self-repairing’ random will accept the path after removing any loops present. The methodology does not make use of any shortest path algorithm, nor does it reduce the number of criteria into a single value. The algorithm is, when reduced to its component parts, very simple and can be seen to consist of an iteration (generations) of three stages; firstly the generation of candidate paths, secondly the application of generic modifiers and finally the selection of optimal paths. The basic outline of the algorithm is provided in Algorithm 4.8.

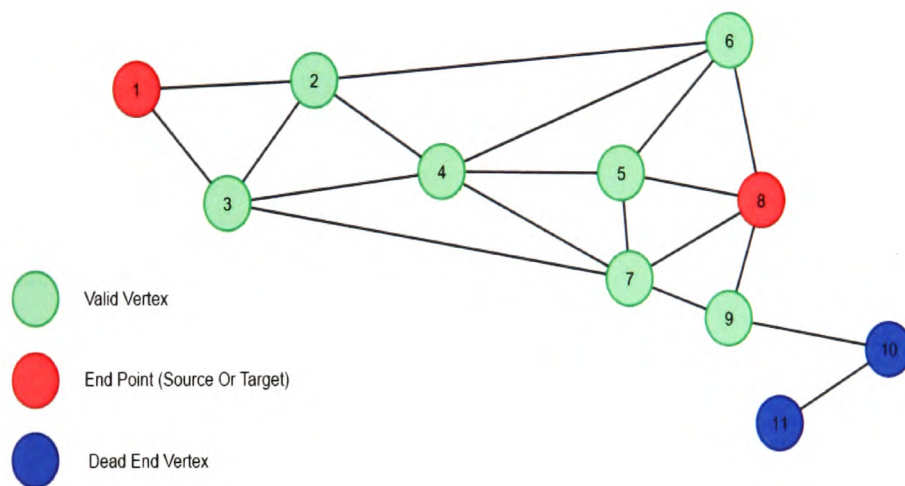


Figure 4.19 Example Graph with 'Dead Ends' Highlighted

Algorithm:	Genetic Algorithm Approach
Input:	S = The ID Of The Source Vertex T = The ID Of The Destination Vertex G = Number Of Generations To Perform P = Size Of The Population $XOverRate$ = Rate Of Crossover $MRate$ = Rate Of Mutation
Output:	$PfAPPROX$ = Approximation Of Optimal Solutions
<hr/> $PfAPPROX = \{ \}$ $CURRENT_POPULATION$ = Generate Population Of $ P $ Solutions Between S And T Using Random Walk $COUNTER = 0$; WHILE ($COUNTER < G$) { $CURRENT_ARCHIVE$ = Extract Pareto Optimal Solutions From $CURRENT_POPULATION$ $RESULTS$ = Perform Genetic Operations On $CURRENT_POPULATION$ Using $XOverRate$, $MRate$ $RESULTS_ARCHIVE$ = Extract Pareto Optimal Solutions From $RESULTS$ $TEMP_ARCHIVE$ = Merge Sets $CURRENT_ARCHIVE, RESULTS, PfAPPROX$ $TEMP_ARCHIVE$ = Extract Pareto Optimal Solutions From $TEMP_ARCHIVE$ $PfAPPROX = TEMP_ARCHIVE$ $CURRENT_POPULATION = TEMP_ARCHIVE$ IF ($ CURRENT_POPULATION < P$) { $NUMBER_OF_PATHS_REQUIRED = P - CURRENT_POPULATION $ NEW_PATHS = Generate $NUMBER_OF_PATHS_REQUIRED$ Using Random Walk $CURRENT_POPULATION = CURRENT_POPULATION + NEW_PATHS$ } $COUNTER = COUNTER + 1$ } } <hr/>	

Algorithm 4.8 Basic Outline of GA Approach

An initial population ($CURRENT_POPULATION$) of unique paths is generated using the random walk technique. At each generation the local Pareto optimal front ($CURRENT_ARCHIVE$) is extracted from the population $CURRENT_POPULATION$. Genetic modifications are undertaken on a copy of that population giving the population ($RESULTS$). A Pareto merge technique is then performed $CURRENT_ARCHIVE$ and the optimal solutions in $RESULTS$. Before a final Pareto extraction is performed giving $TEMP_ARCHIVE$. The resulting population of the final extraction is then expanded with new paths from the random walk methodology. The process is iterated until the

required number of generations has passed. Mooney (2004) highlights the use of terminating conditions when final Pareto population has remained constant for a number of generations. These techniques are not used in this work although they do demonstrate an extremely positive effect on the run time of the algorithm. Practical implementation of the terminating conditions is not undertaken in this study due to their demonstrated success in existing work. Where the test is applied to the K-geodesic approach an estimate of the total number of paths required (given by the equation $(population\ size * generations) * 2$) is produced and stored in an external set. When required by the algorithm a predefined number of solutions are extracted at random from that set.

4.5.1. Evolutionary Operators

This section details the application of the genetic operators used in approach. The implementation of the Genetic Algorithm relies on three genetic operators. The selection of a 'parent' set, the crossover of parents in order to produce variations ('child' paths) and the mutation operator to introduce subtle changes into the population set. Algorithm 4.9 presents an overview of how these algorithms are performed when required by the main process (Algorithm 4.8).

4.5.1.1 Selection Operation

One of the key steps of any evolutionary approach is the selection of a suitable set of solutions from a generation that will proceed to act as the parents for the next generation of the population. Various methods have been suggested as possible selection operators. The various methods include, but are not limited to, tournament selection (Miller and Goldberg 1996), truncation selection (Baker, 1987) and linear ranking selection (Baker 1987). A requirement of the selection mechanism is that it is required to maintain a diverse range in the population set. Goldberg (1989) highlights this requirement whilst also highlighting the need to maintain an adequate level of 'selection pressure'. Selection pressure can be considered the ratio between choosing a

good solution in the space when compared to an average or worse performing solution. A low selection pressure will offer a wide genetic diversity, whilst a high pressure will quickly sacrifice genetic diversity.

Algorithm:	Genetic Operations Algorithm
Input:	<p><i>CURRENT_POPULATION</i> = The Population Prior To Genetic Operations</p> <p><i>XOverRate</i> = Rate Of Crossover</p> <p><i>MRate</i> = Rate Of Mutation</p> <p><i>P</i> = Size Of Population</p>
Output:	<i>UPDATED_POPULATION</i> = Updated Population Following Genetic Operations
<pre> WHILE (<i> CURRENT_POPULATION </i> < <i>P</i>) { <i>PARENT_1</i> = Select Solution From <i>CURRENT_POPULATION</i> at Random <i>PARENT_2</i> = Select Solution From <i>CURRENT_POPULATION</i> at Random WHILE (<i>PARENT_1</i> && <i>PARENT_2</i> Are Equal) <i>PARENT_2</i> = Select Solution From <i>CURRENT_POPULATION</i> at Random <i>R</i> = Select Value Between {0,1.0} At Random IF (<i>R</i> < <i>XOverRate</i>) { Perform Crossover Operation On <i>PARENT_1</i> AND <i>PARENT_2</i> Insert Offspring Into <i>CURRENT_POPULATION</i> Remove Duplicate Solutions From <i>CURRENT_POPULATION</i> } IF (<i>R</i> < <i>MRate</i>) { Perform Mutation Operation Insert Mutated Path Into <i>CURRENT_POPULATION</i> Remove Duplicate Solutions From <i>CURRENT_POPULATION</i> } } <i>UPDATED_POPULATION</i> = <i>CURRENT_POPULATION</i> </pre>	

Algorithm 4.9 Genetic Operators Outline

In the applied algorithm a pair-wise selection method is used. Deb (2001) highlights the feasibility of this approach stating that for almost every crossover operator two solutions are picked from the mating pool at random and some portions of

the solutions are exchanged between the solutions to create a new solution. The use of schemes such as the tournament method would require the condensation of the multi criteria into a single criterion, an effect that the work is trying to avoid.

4.5.1.2 Crossover

The purpose of the crossover procedure is to exchange path information between two paths in order to create a series of new child paths. In an ideal world these two paths will produce better solutions than those offered by the parents, although logically it has to be accepted that this will not always be true.

In the crossover method employed two paths are chosen at random. A list of vertices (other than the source and destination) present in both parents is produced. The contents of the list contain the potential crossover points for the parent paths. A member of the list is selected at random and the sub-paths from the source to the crossover point and the crossover point to the destination are generated and subjected to a recombination procedure to produce the offspring of the parent paths. If the offspring is not present in the general population and the path is valid then it is admitted to the population. Algorithm 4.10 provides an outline of the crossover procedure. Having provided an outline of the procedure the following section provides a practical example of the operation. Figure 4.20 provides an outline of a simple graph, with two selected paths highlighted in Figure 4.21 between vertices 1 and 4 on the graph.

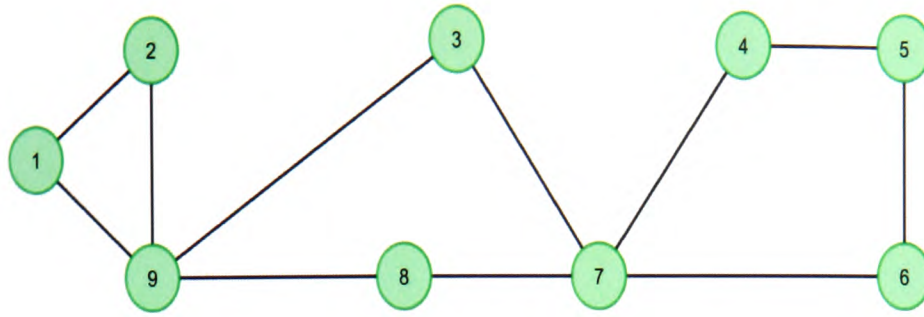


Figure 4.20 Simple Graph for Mutation Operation

Algorithm:	Crossover Operation Algorithm
Input:	<i>PARENT_1</i> = A Path Randomly Selected From Population <i>PARENT_2</i> = A Path Randomly Selected From Population
Output:	<i>CHILD_1</i> = A Path Based On Crossover Of <i>PARENT_1</i> AND <i>PARENT_2</i> <i>CHILD_2</i> = A Path Based On Crossover Of <i>PARENT_1</i> AND <i>PARENT_2</i>
<i>SIZE_1</i> = Length Of <i>PARENT_1</i> <i>SIZE_2</i> = Length Of <i>PARENT_2</i> <i>CROSSOVER_POINTS</i> = { }	
FOR (<i>i</i> = 1 TO <i>SIZE_1</i> -1) FOR (<i>j</i> = 1 TO <i>SIZE_2</i> -1) IF (<i>PARENT_1</i> [<i>i</i>] == <i>PARENT_2</i> [<i>j</i>]) <i>CROSSOVER_POINTS</i> = <i>CROSSOVER_POINTS</i> + <i>PARENT_1</i> [<i>i</i>]	
IF (<i>CROSSOVER_POINTS</i> > 0) { <i>CROSSOVER_POINT</i> = Randomly Select A Value From <i>CROSSOVER_POINTS</i> <i>CHILD_1</i> = Sub Path Of <i>PARENT_1</i> From <i>PARENT_1</i> [0] To <i>CROSSOVER_POINT</i> + Sub Path Of <i>PARENT_2</i> From <i>CROSSOVER_POINT</i> To <i>PARENT_2</i> [<i>PARENT_2</i> -1] <i>CHILD_2</i> = Sub Path Of <i>PARENT_2</i> From <i>PARENT_2</i> [0] To <i>CROSSOVER_POINT</i> + Sub Path Of <i>PARENT_1</i> From <i>CROSSOVER_POINT</i> To <i>PARENT_1</i> [<i>PARENT_1</i> -1] <i>CHILD_1</i> = Perform Remove Cycles Operation On <i>CHILD_1</i> <i>CHILD_2</i> = Perform Remove Cycles Operation On <i>CHILD_2</i> }	
ELSE { // The Crossover Operation Would Not Be Possible <i>CHILD_1</i> = <i>PARENT_1</i> <i>CHILD_2</i> = <i>PARENT_2</i> }	

Algorithm 4.10 Outline of Crossover Procedure

1	2	9	8	7	4	
1	9	3	7	6	5	4

Figure 4.21 Selected Parent Paths Between Vertices 1 and 4

Other than the source and destination two other vertices are present in each path, those being vertex 9 and vertex 7. One of those two vertices is selected at random as the basis for the crossover operation. In the example vertex 7 is selected. Figure 4.22 shows the outcome of the crossover procedure. In place of the original two paths there are now four unique paths between the source and destination.

1	2	9	8	7	4		
1	9	3	7	6	5	4	
1	2	9	8	7	6	5	4
1	9	3	7	4			

Figure 4.22 Results of Crossover Procedure

If the child paths are not present in the general population then they are added to that set. If they are present then they are not added in order to prevent duplicate paths entering the genetic procedure. In Munemoto (1998) a variation to the selected procedure is given where the crossover points are limited to those occupying the same locus on the chromosome. It should be noted that the crossover procedure may result in paths with a lower geodesic value than the parent. This property is seen in Figure 4.22 where a child path is generated with geodesic value of five is generated compared with six and seven for each of the parents.

4.5.1.3 Mutation

The historic approach to the mutation operation has relied on the “flipping” (in a binary context) of a single element in a binary string in order to introduce a small, subtle, change into the chromosome. In the approach to the mutation operator taken here an alternative vertex is introduced into the path to mimic the mutation procedure. A single path is selected with a predefined probability (the mutation rate) from the main population. A vertex (s^N) is then selected at random from the path. The only condition placed on the vertex selection is that it can be neither the start nor the end vertex of the path as such an operation would have no effect and thus be illogical. The graph topology is then searched in order to find any other vertex which have vertex s^{N-1} connected to vertex s^N along with a connection between vertex $s^{N_{\text{replacement}}}$ and vertex s^{N+1} . A list of these potential vertices is generated and stored. One vertex out of the potential vertex list is selected at random and creates a sub-path. The original vertex and the new vertex are switched resulting in a new path. The nature of the graph topology may result in their being no potential replacement vertices, with the probability of there being such a set reliant on the density of the graph. Figure 4.23 presents a view of a graph on which the mutation operator is to be performed. The graph itself is the same as that used in Figure 4.20 reproduced for clarity purposes only.

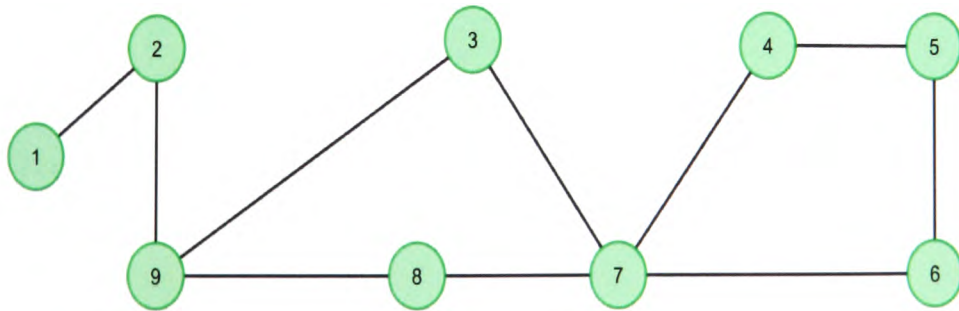


Figure 4.23 Example Graph for the Mutation Operator

Figure 4.24 presents the input to the mutation operator. Figure 4.25 shows the path description vector of the outcome of the mutation operator. In the example vertex eight is selected at random. Vertices 9 and 7 can be seen as predecessor and successor vertices to vertex 8. A search reveals that vertex 3 has the same predecessor and successor vertices. In the outcome of the procedure vertex 8 is replaced by vertex 3. If the output of the algorithm already exists in the population then the path prior to the mutation operation it is discarded.

1	2	9	8	7	4
---	---	---	---	---	---

Figure 4.24 Input PDV Into the Mutation Operator

1	2	9	3	7	4
---	---	---	---	---	---

Figure 4.25 Output PDV From the Mutation Operator

4.6. A PAES Based Approach to the MSPP

The Pareto Archived Evolution Strategy (PAES) is a multi objective based solution that operates on a simple (1+1) local search strategy. An external archive is used to maintain a record of Pareto optimal solutions discovered. In addition, the methodology makes use of functionality that attempts to increase the spread of solutions across the front *PfTRUE*.

The system is initialized with a single randomly generated solution between two vertices using the random walk technique and the “fitness” of that solution is calculated. The initial solution is always added to an external archive. All further solutions produced by the system are derived in some way from this initial path. The system enters a loop which runs until a pre-set number of iterations have been completed or potentially some other terminating condition has been met. During this run variations to the path are introduced which aim to improve the fitness of the solution. The first stage

in the loop process is to develop some form of “mutation” into the initially generated path. A similar approach is taken in all the 1+1 systems such as the Tabu Search or Simulated Annealing. One of three approaches is selected at random; a) a new random path is generated from the start vertex to a randomly selected vertex; b) a new random path is generated from a randomly selected vertex to the end vertex or c) a sub-path from two randomly selected vertices on the path is generated. The process is iterated until a new path is produced (‘new’ as in not the same as the parent). Algorithm 4.15 presents the mutation operator.

Following the application of the path mutation operator a dominance check between the parent(s) and child path(s') is performed. If s dominates s' then no further action is taken until the next iteration is performed. If path s' dominates path s then path s' is added to the external archive of Pareto optimal solutions and made the new path s . Where the two solutions are indifferent the path s' is added to the external archive. If path s' is found to be a Pareto optimal solution then it is accepted as the current solution provided that it occupies a less crowded area than path s . Algorithm 4.11 presents an outline of the PAES and Algorithm 4.12 an outline of the crowding strategy. In the experiments undertaken a maximum size of the archive is not applied given the relatively low number of solutions being considered. Algorithm 4.13 and Algorithm 4.14 are used by the PAES approach to manage the archiving strategy. Algorithm 4.14 is used to identify the location on the Pareto optimal front where a solution can be found and is therefore used to identify crowded areas of the search front while Algorithm 4.13 is used to resize the grid space following the addition to or removal of solutions from the archive of optimal solutions.

Algorithm:	PAES Algorithm
Input:	$MAX_ITERATION$ = Number Of Iterations To Perform MAX_SIZE = Maximum Size Of $PfAPPROX$ $GRID_SIZE$ = The Size Of The Clustering To Use When Determining Crowded Regions.
Output:	$PfAPPROX$ = Set Of Approximations Of $PfTRUE$ / Optimal Solutions
S = Generate A Initial Solution Using Random Walking Evaluate Fitness Of S $PfAPPROX = PfAPPROX + S$ $CURRENT_ITERATION = 1$ WHILE ($CURRENT_ITERATION < MAX_ITERATION$) { S' = Perform Perturb Mechanism On Solution S Evaluate Fitness Of S' IF (S Dominates S') Continue // There Is Nothing Left To Do This Iteration ELSE IF (S' Dominates S) { $S = S'$ Perform Grid Updating Algorithm $PfAPPROX = PfAPPROX + S$ $PfAPPROX$ = Extract Optimal Solutions From $PfAPPROX$ } ELSE IF (S' Is Dominated By Any Member Of $PfAPPROX$) Continue // There Is Nothing Left To Do This Iteration ELSE { S = Apply Archiving Strategy($S, S', PfAPPROX, MAX_SIZE, GRID_SIZE$) Perform Grid Updating Algorithm $PfAPPROX = PfAPPROX + S$ $PfAPPROX$ = Extract Optimal Solutions From $PfAPPROX$ } $CURRENT_ITERATION = CURRENT_ITERATION + 1$ }	

Algorithm 4.11 Outline of the PAES Algorithm

Algorithm:	PAES Archiving Strategy Archive Algorithm
Input:	MAX_SIZE = Maximum Number Of Optimal Solutions To Keep C = Current Solution M = Updated Solution $PfAPPROX$ = Set Of Current Optimal Solutions
Output:	$PfAPPROX'$ = Updated Set Of Optimal Solutions
$REPLACEMENT_INDEX = -1$ IF ($ PfAPPROX < MAX_SIZE$) $PfAPPROX' = PfAPPROX + M$ Exit // No Need To Do Anything Else IF (C Is Not Dominated By Any Solution On $PfAPPROX$) { M_LOC = Identify Grid Location Using Find Location Operation M_COUNT = Count Solutions In Grid Location M_LOC FOR ($i = 0$ TO $ PfAPPROX $) { $TEMP$ = Identify Grid Location Of $PfAPPROX[i]$ Using Find Location Operation $TEMP_COUNT$ = Count Solutions In Grid Location $TEMP$ IF ($M_COUNT > TEMP_COUNT$) $REPLACEMENT_INDEX = i$ } IF ($REPLACEMENT_INDEX > -1$) $PfAPPROX[REPLACEMENT_INDEX] = M$ $C = M$ }	
$PfAPPROX' = PfAPPROX$	

Algorithm 4.12 Outline of the PAES Archiving Strategy

Algorithm:	Update Grid Algorithm
Input:	<p><i>PATH</i> = The Find For Which The Location Is Sought</p> <p><i>DEPTH</i> = The Number Of Cells Into Which The Grid Has Been Divided</p> <p><i>OBJECTIVES</i> = Number Of Objectives Being Considered</p> <p><i>PfAPPROX</i> = Set Of Optimal Solutions</p>
Output:	<p><i>OFFSETS</i> = { }</p> <p><i>LARGEST</i> = { }</p> <p>FOR (<i>i</i> = 0 TO <i>OBJECTIVES</i>)</p> <p> <i>OFFSETS</i>[<i>i</i>] = Get Smallest Value Of Each Objective</p> <p> <i>LARGEST</i>[<i>i</i>] = Get Largest Values Of Each Objective</p> <p>FOR (<i>j</i> = 0 TO <i>OBJECTIVES</i>)</p> <p> IF (<i>PATH.Cost</i>[<i>j</i>] < <i>OFFSET</i>[<i>j</i>])</p> <p> <i>OFFSET</i>[<i>k</i>] = <i>PATH.Cost</i>[<i>k</i>]</p> <p> IF (<i>PATH.Cost</i>[<i>j</i>] > <i>LARGEST</i>[<i>j</i>])</p> <p> <i>LARGEST</i>[<i>j</i>] = <i>PATH.Cost</i>[<i>j</i>]</p> <p><i>SSE</i> = Calculate Difference Between Minima and Maxima Of Each Objective</p> <p>IF (<i>SSE</i> > 0.1) // 10 %</p> <p>{</p> <p> Renormalize The Space By 20%</p> <p> Recalculate The Locations Of Each Solution In <i>PfAPPROX</i> Using Find Location Procedure</p> <p>}</p> <p>Identify the Location Of <i>PATH</i> in Grid Space Using Find Location Procedure</p>

Algorithm 4.13 PAES Update Grid Algorithm

Algorithm:	Find Location Algorithm
Input:	<i>PATH</i> = The Find For Which The Location Is Sought <i>PfAPPROX</i> = Set Of Optimal Solutions <i>DEPTH</i> = Number Of Cells Into Which The Grid Has Been Separated <i>OBJECTIVES</i> = Number Of Criteria Being Considered
Output:	<i>LOCATION</i> = The Location In The Grid Space
<hr/> <i>INC</i> = { } <i>WIDTH</i> = { } <i>N</i> = 1 FOR (<i>i</i> = 0 TO <i>OBJECTIVES</i>) <i>INC</i> [<i>i</i>] = <i>N</i> <i>N</i> = <i>N</i> * 2 <i>WIDTH</i> [<i>i</i>] = Range For This Objective Based On <i>PfAPPROX</i> // Largest Value – Smallest Value FOR (<i>j</i> = 0 TO <i>DEPTH</i>) FOR (<i>k</i> = 0 TO <i>OBJECTIVES</i>) IF (<i>PATH.Cost</i> [<i>k</i>] < <i>WIDTH</i> [<i>k</i>] / 2 + Smallest Value For This Objective) <i>LOCATION</i> = <i>LOCATION</i> + <i>INC</i> [<i>i</i>] ELSE Increase The Offset For This Location FOR (<i>l</i> = 0 TO <i>OBJECTIVES</i>) <i>INC</i> [<i>l</i>] = <i>INC</i> [<i>l</i>] * (2 * <i>OBJECTIVES</i>) <i>WIDTH</i> [<i>i</i>] = <i>WIDTH</i> [<i>i</i>] / 2	

Algorithm 4.14 PAES Find Location Algorithm

Algorithm:	Mutation Algorithm
Input:	$PATH$ = The Current Solution $G = (V , E)$ = The Graph Which Is Being Analysed
Output:	$PATH'$ = The Mutated Path
<hr/> $COUNT = PATH - 1$ $M = \text{Select At Random } \{1, 2, 3\}$ IF ($M == 1$) { $MUTATION_POINT = \text{Select A Point On } PATH > PATH[0] \text{ And } < PATH[PATH -1] \text{ At Random}$ $SP = \text{Generate A Random Path From } PATH[0] \text{ To } MUTATION_POINT$ $PATH' = SP + (PATH[MUTATION_POINT], PATH[PATH -1])$ } ELSE IF ($M == 2$) { $MUTATION_POINT = \text{Select A Point On } PATH > PATH[0] \text{ And } < PATH[PATH -1] \text{ At Random}$ $SP = \text{Generate A Random Path From } PATH[MUTATION_POINT] \text{ To } PATH[PATH -1]$ $PATH' = (PATH[0], PATH[MUTATION_POINT]) + SP$ } ELSE IF ($M == 3$) { $M = \text{Select At Random } \{1: 2\}$ IF ($M == 1$) { $POINT_A = \text{Select A Point On } PATH > PATH[0] \text{ And } < PATH[PATH -1] \text{ At Random}$ $POINT_B = \text{Select A Point On } PATH > PATH[0] \text{ And } < PATH[PATH -1] \text{ At Random}$ $SP = \text{Generate A Random Path Between } POINT_A \text{ And } POINT_B$ $PATH' = (PATH[0], PATH[POINT_A]) + SP + (PATH[POINT_B], PATH[PATH -1])$ } ELSE IF ($M == 2$) { $POINT = \text{Select A Point On } PATH > PATH[0] \text{ And } < PATH[PATH -1] \text{ At Random}$ $SP_A = \text{Generate A Random Path Between } PATH[0] \text{ And } POINT$ $SP_B = \text{Generate A Random Path Between } POINT \text{ And } PATH[PATH -1]$ $PATH' = SP_A + SP_B$ } } } <hr/>	

Algorithm 4.15 1+1 Mutation Operator

4.7. Tabu Search Algorithm

In this implementation of the Tabu Search algorithm two 'Tabu' lists are employed. The first of these is considered to be 'long term memory' based solutions and consists of those solutions which are currently on the Pareto front ($PfAPPROX$). The second Tabu memory set is shorter term memory and consists of a set of solutions which do not require re-examination at this point in time. As the algorithm advances the

Tabu list acts a *FIFO* (First In First Out) list. As new Tabu solutions are added, older Tabu solutions are taken from the list. It should be noted that the Tabu list is the set of solutions on *PfAPPROX* and a secondary list of solutions recently considered. Solutions present on *PfAPPROX* are never removed from the list of Tabu solutions, unless those solutions are found to be dominated in some way.

The strength of Tabu Search lies within its use of short-term memory to store previous moves. Unless recent moves are rendered inadmissible (or ‘Tabu’) the search process could make a ‘best’ (non-improving) move away from a local optimum and then fall back into the ‘local best’ when later moves are made. Algorithm 4.16 presents the methodology of the Tabu Search employed in this work. The algorithm first generates an initial path (*CURRENT*) using the random walk technique. The initial path is added to both the external archive and the Tabu list. A neighbour (*MUTATED*) of that path is generated using the methodology outlined in Algorithm 4.15. If *MUTATED* is not dominated by the current solution then it made the current solution and added to the external archive. In future iterations comparisons will be carried out on path *MUTATED* or successive versions of that path. Where the solutions are indifferent or *CURRENT* dominates *MUTATED* solution *MUTATED* will be accepted as current only when it allows a move that can be considered ‘closer’ to the Pareto optimal front as suggested by the front *PfAPPROX*. As that mechanism alone may cause the methodology to become ‘stuck’, the methodology randomly generates an entirely new path when an indifferent solution is not closer to *PfAPPROX*, i.e. if *CURRENT* is a member of *PfAPPROX* then *P'* may never be closer. Therefore *CURRENT* is randomly re-initialized following the distance check. Algorithm 4.17 outlines the distance calculation method. Regardless of whether or not a solution is accepted as *CURRENT* it is added to the Tabu list. The Tabu list is ‘pruned’ in order to ensure that its maximum length is not exceeded before the next iteration is performed. If a solution is present on the Tabu list then it is not considered.

Algorithm:	Tabu Search Algorithm
Input:	S = The Source Vertex T = The Target Vertex $G = (V , E)$ = The Graph Which Is Being Analysed $TABU_SIZE$ = The Maximum Size Of The Tabu List $MAX_ITERATIONS$ = The Number Of Iterations To Perform
Output:	$PfAPPROX$ = The Optimal Set Of Solutions
<hr/> $PfAPPROX = \{ \}$ $TABU_LIST = \{ \}$ $CURRENT$ = Generate A Path Between S And T At Random Using Random Walk $PfAPPROX = PfAPPROX + CURRENT$ $ITERATION = 0$ WHILE ($ITERATION < MAX_ITERATIONS$) { $MUTATED$ = Perform Mutation Algorithm On $CURRENT$ WHILE ($MUTATED$ Is Tabu) // Present On $PfAPPROX$ Or $TABU_LIST$ $MUTATED$ = Perform Mutation Algorithm On $CURRENT$ $TABU_LIST = TABU_LIST + MUTATED$ IF ($MUTATED$ Dominates Any Member Of $PfAPPROX$) { $CURRENT = MUTATED$ $PfAPPROX = PfAPPROX + CURRENT$ $PfAPPROX$ = Extract Optimal Solutions From $PfAPPROX$ } ELSE { IF ($MUTATED$ Dominates $CURRENT$) { $CURRENT = MUTATED$ $PfAPPROX = PfAPPROX + CURRENT$ $PfAPPROX$ = Extract Optimal Solutions From $PfAPPROX$ } ELSE IF ($MUTATED$ And $CURRENT$ Are Indifferent) { $DCurrent$ = Calculate Average Distance From $PfAPPROX$ To $CURRENT$ $DMutated$ = Calculate Average Distance From $PfAPPROX$ To $MUTATED$ IF ($DMutated < DCurrent$) { $CURRENT = MUTATED$ $PfAPPROX = PfAPPROX + CURRENT$ $PfAPPROX$ = Extract Optimal Solutions From $PfAPPROX$ } ELSE IF ($DMutated == DCurrent$) { R = Random Value In $\{0.0, 1.0\}$ IF ($R < 0.5$) { $CURRENT$ = Generate A Path New Path Between S And T $PfAPPROX = PfAPPROX + CURRENT$ $PfAPPROX$ = Extract Optimal Solutions From $PfAPPROX$ } } } } $ITERATION = ITERATION + 1$ IF ($ TABU_LIST > TABU_SIZE$) Prune Oldest Solution From $TABU_LIST$ } } <hr/>	

Algorithm 4.16 Outline of the Tabu Search

Algorithm:	Distance Calculation Algorithm
Input:	$PfAPPROX$ = Current Set Of Pareto Optimal Approximations $PATH$ = A Path Representing A Possible Solution
Output:	AV_D = Average Distance Of A Solution To $PfAPPROX$
<hr/> $TOTAL_DISTANCE = 0$ $SIZE = PfAPPROX $ $DISTANCES = \{ \}$ FOREACH ($MEMBER$ Of $PfAPPROX$) { $CRITERIA_COUNT$ = Number Of Criteria In ($PATH$ && $MEMBER$) $D = 0$ $DELTA = \{ \}$ FOR ($i = 0$ TO $CRITERIA_COUNT$) $DELTA[i] = MEMBER.Costs[i] = PATH.Costs[i]$ FOR ($j = 0$ TO $CRITERIA_COUNT$) $DELTA[j] = DELTA[j] * DELTA[j]$ $TOTAL = 0$ FOR ($k = 0$ TO $CRITERIA_COUNT$) $TOTAL = TOTAL + DELTA[j]$ $DISTANCES[MEMBER] = SQRT(TOTAL)$ } FOR ($l = 0$ TO $ DISTANCES $) $TOTAL_DISTANCE = TOTAL_DISTANCE + DISTANCES[l]$ $AV_D = TOTAL_DISTANCE / DISTANCES $	

Algorithm 4.17 Distance Calculation for Indifferent Solutions

4.8. Simulated Annealing Algorithm

In this section an algorithm is introduced that aims to solve the MSPP based on a Simulated Annealing technique. The algorithm shares the basic properties of the work developed in Smith (2006) in which a Simulated Annealing approach is introduced with the basic property that the algorithm does not reduce the multi objective problem into a single criterion methodology through the use of weightings and an additive process for each criterion. The majority of schemes that adapt Simulated Annealing to more than

one objective (Serafini, 1994, Ulungu *et al.*, 1999, Czyzak and Jaszekiewicz, 1998, Nam and Park, 2000, Hapke *et al.*, 2000, Suppakitnarm *et al.*, 2000, Tuytens *et al.*, 2003) have focused on a single solution, and determining the quality of perturbations to that single solution using a weighted sum of objectives.

The archive is initialised with the initial feasible path (*CURRENT*). The algorithm is also initialized with a starting temperature, cool rate and closing temperature. The initial path is added to an external archive. At each stage of the annealing process the current solution *CURRENT* is perturbed using the methodology presented in Algorithm 4.15 to form the proposed solution *MUTATED*. If solution *MUTATED* dominates solution *CURRENT* then it is accepted as the new *CURRENT* and added to the external archive. Dominated solutions are removed from the external archive. If the solution is indifferent then the solution is accepted based on its distance to the front *PfTRUE*. Unlike the Tabu Search where solutions that are further away from the front *PfTRUE* are discarded in the Simulated Annealing approach they are accepted based on an annealing probability. Algorithm 4.19 presents an outline of the Simulated Annealing approach used in this work. Algorithm 4.18 is used to calculate the probability of acceptance of an indifferent solution.

Algorithm:	Probability Of Acceptance Algorithm
Input:	$DCurrent$ = Average Distance Of The Current Solution To <i>PfAPPROX</i> $DMutated$ = Average Distance Of The Mutated Solution To <i>PfAPPROX</i> $TEMPRATURE$ = The Current Temperature Of The Annealing Process
Output:	P = The Probability Of Accepting <i>DMutated</i> As The Current Solution
$DELTA = DCurrent - DMutated$ $TEMP = DELTA / TEMPRATURE$ $P = EXP(TEMP)$	

Algorithm 4.18 Probability of Acceptance Algorithm

Algorithm:	Simulated Annealing Algorithm
Input:	S = The Source Vertex T = The Target Vertex $G = (V , E)$ = The Graph Which Is Being Analysed $STARTING_TEMPRATURE$ = The Starting Temperature $COOLING_RATE$ = How Quickly The Temperature Cools $CLOSING_TEMPRATURE$ = When Does The Algorithm Terminate
Output:	$PfAPPROX$ = The Optimal Set Of Solutions
<hr/> $PfAPPROX = \{ \}$ $TABU_LIST = \{ \}$ $CURRENT$ = Generate A Path Between S And T At Random Using Random Walk $PfAPPROX = PfAPPROX + CURRENT$ $TEMPRATURE = STARTING_TEMPRATURE$ WHILE ($TEMPRATURE > CLOSING_TEMPRATURE$) { $MUTATED$ = Perform Mutation Algorithm On $CURRENT$ IF ($MUTATED$ Dominates Any Member Of $PfAPPROX$) { $CURRENT = MUTATED$ $PfAPPROX = PfAPPROX + CURRENT$ $PfAPPROX$ = Extract Optimal Solutions From $PfAPPROX$ } ELSE { IF ($MUTATED$ Dominates $CURRENT$) $CURRENT = MUTATED$ $PfAPPROX = PfAPPROX + CURRENT$ $PfAPPROX$ = Extract Optimal Solutions From $PfAPPROX$ ELSE IF ($MUTATED$ And $CURRENT$ Are Indifferent) { $DCurrent$ = Calculate Average Distance From $PfAPPROX$ To $CURRENT$ $DMutated$ = Calculate Average Distance From $PfAPPROX$ To $MUTATED$ IF ($DMutated < DCurrent$) $CURRENT = MUTATED$ $PfAPPROX = PfAPPROX + CURRENT$ $PfAPPROX$ = Extract Optimal Solutions From $PfAPPROX$ ELSE // Difference Between Tabu Search – Not Only Indifference Considered { P = Calculate Probability Of Acceptance R = Random Value In $\{0.0, 1.0\}$ IF ($R < P$) $CURRENT = MUTATED$ $PfAPPROX = PfAPPROX + CURRENT$ $PfAPPROX$ = Extract Optimal Solutions From $PfAPPROX$ } } } $TEMPRATURE = TEMPRATURE * COOLING_RATE$ } }	

Algorithm 4.19 Outline of Simulated Annealing Algorithm

4.9. Dominance and extraction of *PfAPPROX*

Chapter Two of this thesis introduced the concept of Pareto optimality and dominance through a worked example. In this section the method the algorithms used are introduced briefly. Algorithm 4.20 represents the methodology used to detect the dominance relationship between two solutions while Algorithm 4.21 extends Algorithm 4.20 into the extraction of the Pareto optimal set.

Algorithm:	Dominance Check Algorithm
Input:	CA = Costs Of Candidate Solution A CB = Costs Of Candidate Solution B
Output:	True If Solution A Dominates Solution B, False If Not
$C = CA $ $D = \text{True}$ $i = 0$ WHILE $((i < C) \ \&\& \ D)$ { $D = (D \ \&\& \ (CA[i] \geq CB[i]))$ $i = i + 1$ }	

Algorithm 4.20 Dominance Check Algorithm

Algorithm:	Extraction Of Optimal Solutions Algorithm
Input:	P = Population Of Solutions
Output:	$PfAPPROX$ = Non Dominated Solutions From Population P
$PfAPPROX = \{ \}$ FOREACH (Path A in P) { FOREACH (Path B in P) { IF ($A \ \&\& \ B$ Are Equal) Continue ELSE { IF (A Dominates B) Mark B As Dominated } } } FOREACH (Path C in P) { If (C Is Not Marked As Dominated) $PfAPPROX = PfAPPROX + C$ }	

Algorithm 4.21 Extraction of Pareto Optimal Front

4.10. Support Algorithms

A number of support algorithms are used internally to the algorithms. These include methods to check if any two paths are equal, if a population contains a solution and other associated functionality. These are described here briefly. Given their support nature exact detail is not included beyond the high level pseudo code descriptions provided.

Algorithm 4.22 is used to determine if two paths are equal. A simple comparison will not identify equality between two paths and so additional checks are required. The procedure first counts the number of vertices in the two paths. If the length values are not equal then the paths themselves will not be equal and so the procedure exits indicating that fact. If the two paths do have the same number of vertices iteration over the cost values is performed. If all the costs are equal the path is assumed to be equal. If any cost is not the same between the two paths the paths will not be considered equal and the procedure is exited returning false. In order to reach this stage in the equality check the two paths will have to have the same lengths and criteria costs. The procedure therefore iterates the path descriptions themselves. If any two vertices at the same locus on the path are unequal the paths are considered unequal. If all the vertices at each locus are equal the paths are considered equal.

Algorithm 4.23 is used to determine if a population such as a generation of the Genetic Algorithm or the tabu list in the Tabu Search Algorithm already contains a path. The method iterates through each member perform Algorithm 4.22 to determine if the paths are equal. If at any point the method receives a value of true from that algorithm then it does already contain a path.

Algorithm 4.24 is an extension of Algorithm 4.21 that returns a set of non dominated solutions from the provided set. Algorithm 4.24 performs the same basic functionality with a series of minor variations. The method is passed a single candidate path in addition to a populated set of solutions. If the passed candidate solution would dominate any member of the provided population (i.e. it would be a member of that population) then the method returns true. Otherwise the method return false. Algorithm 4.25 is used to identify the costs associated with a provided path. It looks up each costs for each criteria in the graph for each edge in the graph calculating a total sum for each criteria.

Algorithm:	Path Equality Algorithm
Input:	A = Path Between Two Vertices B = Path Between Two Vertices
Output:	True If The Paths A And B Are Equal, False If Not
<pre> If ($A \neq B$) Return <i>False</i> Else { $COST_COUNT = A.Costs$ $ARE_EQUAL = True$ FOR ($i = 0$ TO $COST_COUNT$) { IF ($A.Costs[i] \neq B.Costs[i]$) $ARE_EQUAL = False$ } IF ($!ARE_EQUAL$) Return <i>False</i> FOR ($j = 0$ TO A) { IF ($A[j] \neq B[j]$) Return <i>False</i> } Return <i>True</i> } </pre>	

Algorithm 4.22 Path Equality Algorithm

Algorithm:	Population Contains Algorithm
Input:	$PfAPPROX$ = A Population Of Solution $PATH$ = Path Between Two Vertices
Output:	True If The $PfAPPROX$ Contains $PATH$, False If Not
FOREACH (Path P in $PfAPPROX$)	
{	
IF (P && $PATH$ Are Equal)	
Return True	
}	
Return False	

Algorithm 4.23 Population Contains Algorithm

Algorithm:	Simple Dominates Algorithm
Input:	P = Population Of Solutions $PATH$ = A Solution Being Check For Dominance Against P
Output:	True If The Solution
$PfAPPROX = P$	
$PfAPPROX = PfAPPROX + PATH$	
$PfAPPROXCopy = \{ \}$	
FOREACH (Path A in $PfAPPROX$)	
{	
FOREACH (Path B in $PfAPPROX$)	
{	
IF (A && B Are Equal)	
Continue	
ELSE	
{	
IF (A Dominates B)	
Mark B As Dominated	
}	
}	
}	
FOREACH (Path C in $PfAPPROX$)	
{	
If (C Is Not Marked As Dominated)	
$PfAPPROXCopy = PfAPPROXCopy + C$	
}	
IF ($PfAPPROXCopy$ Contains $PATH$)	
Return True	
ELSE	
Return False	

Algorithm 4.24 Simple Population Domination Check

Algorithm:	Calculate Path Description Vector Algorithm
Input:	<i>PATH</i> = Path Between Two Vertices $G = (V , E)$ = The Graph Being Analysed C = Number Of Costs Associated With edge $e(i,j) \in E$
Output:	<i>PATH</i> With Path Description Vector Of C Costs
<i>PATH.Costs</i> = { } <i>SIZE</i> = <i>PATH</i> FOR ($l = 0$ To <i>SIZE</i> -1) { $i = \text{PATH}[l]$ $j = \text{PATH}[l+1]$ Edge $e(i,j) = G(e(i,j))$ FOR ($k = 0$ TO C) $\text{PATH.COSTS}[k] = e(i,j).\text{Costs}[k]$ } 	

Algorithm 4.25 Calculation of the Path Description Vector

4.11. Chapter Summary

This chapter outlined the various algorithms developed or used in the study for the solution to the MSPP. The data sets used to test the algorithms has been introduced, together with a rationale provided for their inclusion. The test data used in the study consists of both real world and synthetic graphs of a variety of sizes, ranging from what may be considered to be very small graphs through to larger graphs. The application of both data sets should provide a robust challenge to the techniques with each exhibiting differing qualities, such as real world graphs having the general property of a higher portion of links with only two connected vertices. Also introduced were mechanisms that provide robust approximations of the *PfTRUE*, against which the outputs (during the experimental phase of this work described in the following two chapters) of the algorithms will be compared.

A review of the literature of alternatives to the Evolutionary/Genetic Algorithm approach to multi criteria optimisation has been undertaken in Chapter 3. The Methods developed as part of this study and described in this chapter attempt to build on the most promising aspects of those algorithms, where particular emphasis' has been placed on avoiding reducing the multi criteria optimisation process to a single criteria process through the application of weighting to each of the criteria.

As has been highlighted frequently elsewhere in this work, no existing examples of meta-heuristic methods other than the Genetic Algorithms can be seen when applied to the solution of the MSPP. In that regard, the algorithms developed are certainly novel. In the next chapter each of the developed approaches will be compared to the Genetic Algorithm approach and existing algorithmic methods on the test data.

Chapter 5: Runtimes and Scalability

5. Runtimes and Scalability

The current chapter reviews the performance of the algorithms. The chapter first considers the Dijkstra shortest path algorithm applied to multiple criteria in order to identify optimal paths in each criterion. The chapter then considers other algorithmic solutions to the MSPP in the form of the Skriver and Andersen algorithm before reviewing the effectiveness of the Climaco and Martins algorithm. Attention then turns to presenting an overview of the heuristics developed in terms of runtime and quality. At the end of the chapter an overview is provided to highlight scenarios where the application of any particular algorithm may be considered appropriate.

5.1. Algorithmic Solutions to the MSPP

Three algorithmic methods are reviewed in this section. The first is arguably the most primitive and merely involves applying the Dijkstra shortest path algorithm to a series of independent criteria. Attention then turns towards those methods that model the Pareto optimal front in its entirety (generating methods). The performance of the algorithm outlined in Skriver and Anderson (2000) is considered alongside the path/tree based strategy introduced in Climaco and Martins (1982). The generating algorithms are tested against a series of random and real world graphs.

5.1.1. Dijkstra Algorithm Using Multiple Criteria

In this section the performance of Dijkstra's shortest path algorithm when applied to multiple criteria is reviewed. The experiment effectively runs the Dijkstra shortest path algorithm D times where D would be equal to the number of criteria under consideration. The experiment was repeated 1000 times. Figure 5.1 presents the results graphically along with the base line figure for Dijkstra shortest path algorithm. The base

line is the performance of the Dijkstra shortest path algorithm when calculating a single shortest path or when $D = 1$ For the purposes of the experiment considering multiple criteria $D = 4$.

The Dijkstra shortest path algorithm applied to multiple criteria offers limited insight into the front $PfTRUE$ as the methodology merely calculates the extreme points of the Pareto optimal front and as such may offer little value for a large number of applications. However, referring to the descriptions in Chapter 1 as to the value of route planning and how route selection is performed then the technique may hold some, albeit limited, value. A similar view is presented in Hallam *et al* (2001 p.134), who suggest: “it may be of little practical use to find all Pareto-optimal paths. For many applications it is sufficient to know at most one Pareto-optimal path in $P(s, t)$ for each minimal cost vector”. The test demonstrates that the extreme Pareto points can be identified quickly where it is considered of value.

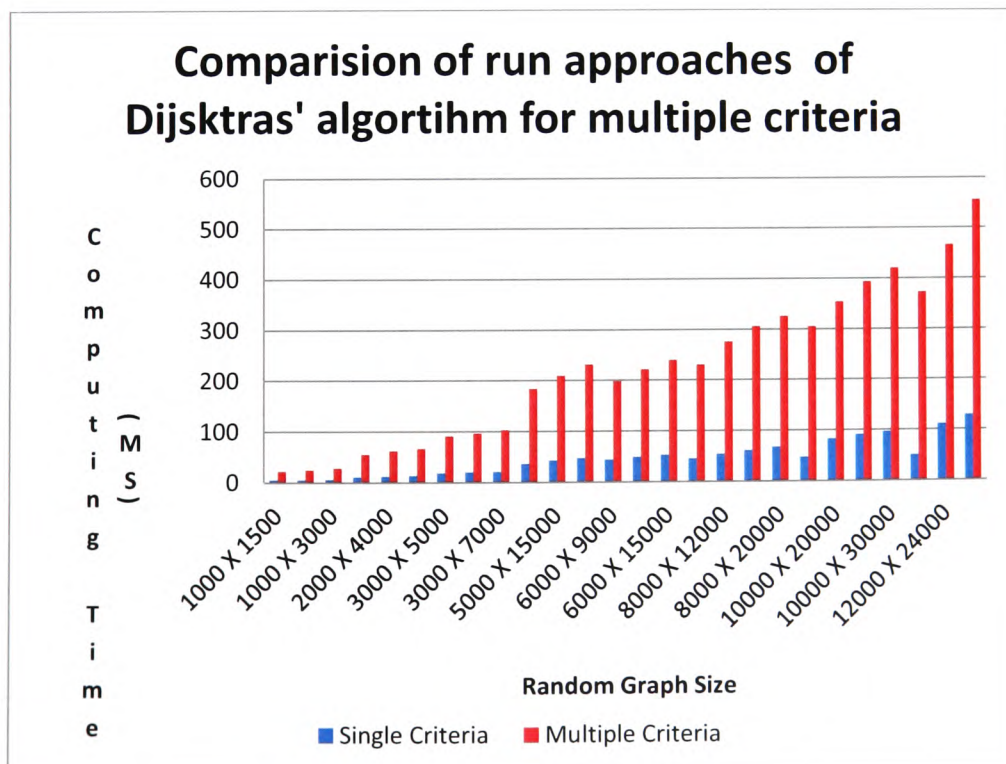


Figure 5.1 Graph of Multi Criteria Dijkstra Approach Runtimes

5.1.2. Skriver and Anderson's Algorithm

At this point the runtimes of the Skriver and Anderson (2000) label-correcting algorithm are presented when applied to a subset of real world and random graphs. The algorithm aims to generate the complete set $PfTRUE$. Table 5.1 presents the runtimes of the algorithm when applied to real world graphs while Table 5.2 presents the results of the algorithm on random graphs. The runtimes of Table 5.1 and Table 5.2 are presented in chart form below in Figure 5.2 and Figure 5.3 respectively. The tests demonstrate how increasing the graph size has a substantial effect on algorithm runtime. This appears to be due to the requirement for edge costs to be sorted for each criterion in order for the over-take mechanism suggested by Skriver and Andersen to be effective. The experiment is performed between 100 randomly selected vertex pairings. The results of the experimentation provided in Table 5.1 and Table 5.2 should be used with a certain degree of caution due to differences in the size of the graphs with the smallest real world graph being much larger than the smallest random graph. However, the general structure of the graph appears to have a substantial effect on the runtime of the Skriver and Andersen algorithm. Taking as an example the smallest of the real world graphs, Cardiff 500 has 2288 edges and 1076 vertices. The Skriver and Andersen algorithm takes an average of 101.67 seconds to complete a processing run. In comparison the largest random graph 1000 X 2000 completes a processing run 132 seconds. It is believed that the difference in runtime is a due to the difference in the makeup of the graph. The real world graphs exhibit the property of having a large number of vertices with a low degree value, forming long isolated strings. The method of operation of the algorithm appears to often delete such edges early on in the processing run having a positive effect on the run time of the algorithm.

Graph	Seconds
Cardiff 500	101.67
Cardiff 750	221.45
Cardiff 1000	331.23
Cardiff 1500	642.27

Table 5.1 Runtime of S&A Algorithm on Real World Graphs

Graph	Seconds
100 X 200	1.6
200 X 400	11.6
500 X 1000	81
750 X 1500	108
1000 X 2000	132

Table 5.2 Runtime of S&A Algorithm on Random Graphs

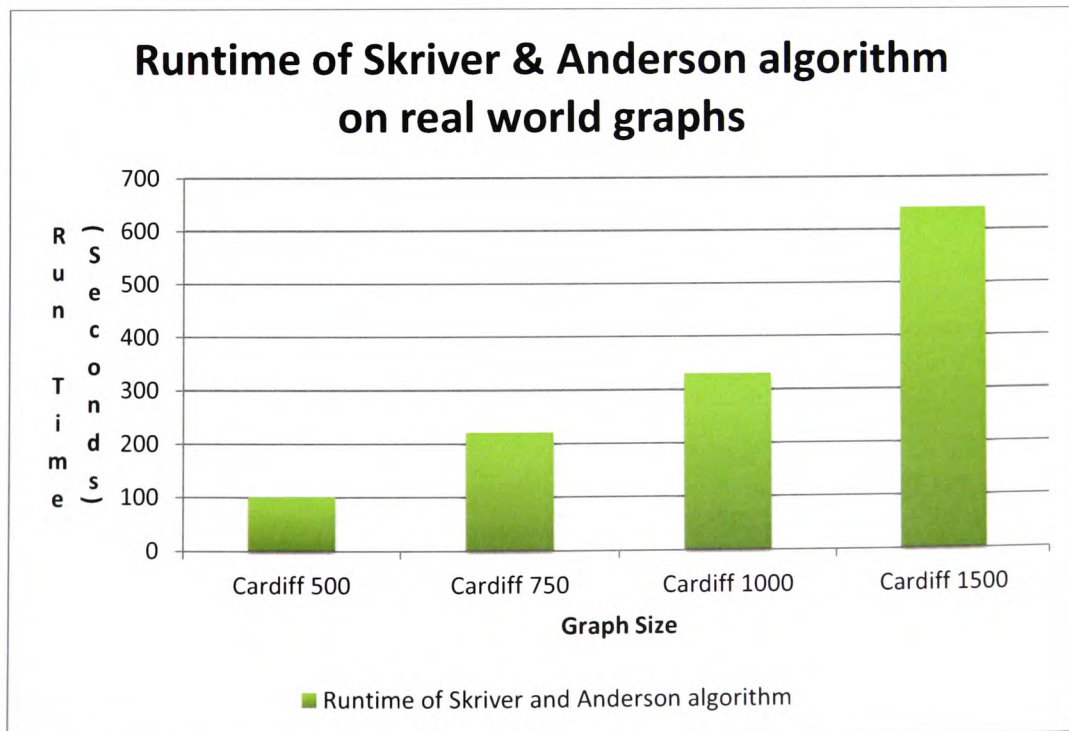


Figure 5.2 Runtime of S&A Algorithm on Real World Graphs

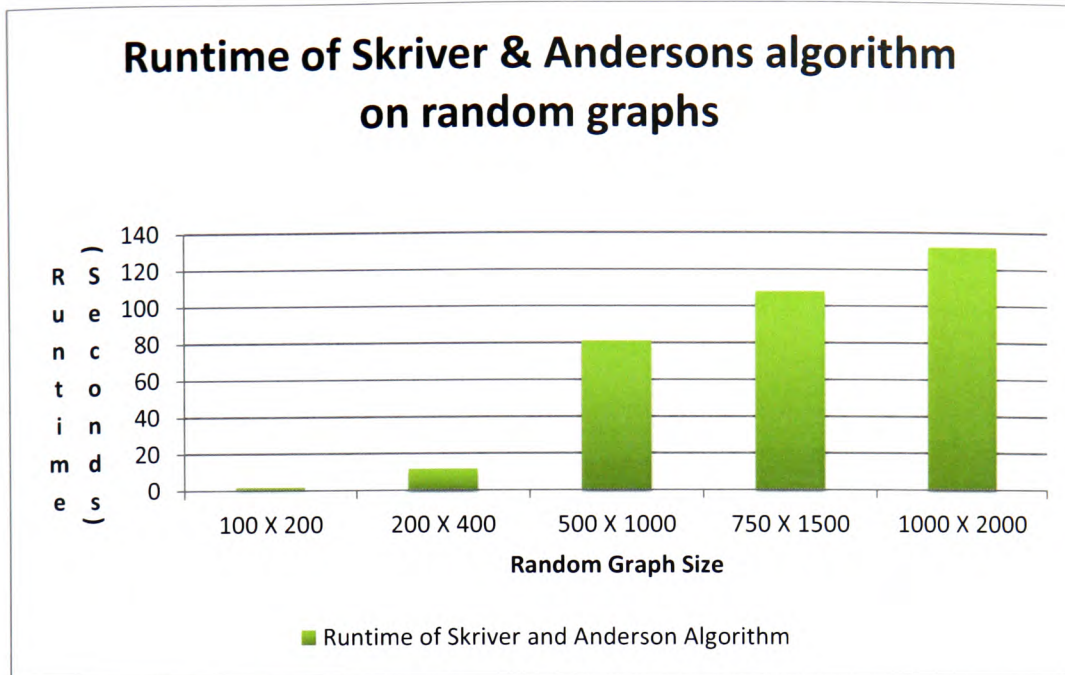


Figure 5.3 Runtime of S&A on Random Graphs

5.1.3. Climaco and Martins' Algorithm

In the following experiment, an implementation of Climaco and Martins MSPP algorithm is tested. Skriver and Anderson (2000) state that the labelling methodology should theoretically always outperform path or tree approaches as seen in the Climaco and Martins approach. The algorithm works by firstly calculating the shortest path (P^x) between two vertices on a single criteria. It then calculates the cost of traversing the same path using the second criteria (P^k). A K-shortest path algorithm is then used to iteratively optimise the second criteria until the cost of P^{k2} exceeds the cost of traversing P^k . When that condition has been met the algorithm will have discovered the complete set $PfTRUE$.

In Raith and Ehrgott (2009) a study on both real world (road) graphs and random graphs is presented. Several scenarios are identified where the proposition of Skriver and Anderson (2000) is demonstrated not to be true with a variation to the K shortest path algorithm presenting results that in many cases show the tree based

approach outperforming the results of the Skriver and Anderson approach when applied to real world graphs. In their work however Raith and Ehrgott also find several cases when the Skriver and Anderson approach dominates (outperforms by a factor of over 10) the tree-based approach used by Raith and Ehrgott. In the following experiment a study into the runtime of the Climaco and Martins approach is presented together with a hypothesis for the performance difference. An attempt is made to illustrate why vertex selection plays a vital role in the success of the Climaco and Martins approach. The experiment attempts to optimise the distance and travel time of real world networks. In the data preparation stage vertex pairs are selected and split into two separate groupings. The first grouping contains a series of vertices where the delta of the distance and time between the optimal paths on each criterion is small; the second grouping consists of vertex pairings where the delta value is large.

For each vertex pairing two shortest path analysis are performed. The first optimises the distance values between the source and destination vertex. The second optimises the travel times between the same vertices. Following the calculation of each shortest path the corresponding cost is calculated for traversing the same path using the other criteria. Delta values are generated between the two shortest paths for each criterion. Two groups consisting of 100 vertex pairings are selected. The first consists of those solutions where the delta values are smaller, with an added filter applied to ensure that the distance delta is greater than 350 meters and the travel time delta greater than 20 seconds. The second set consists of the 100 vertex pairings with the highest deltas between the distance and travel time.

The first of the columns in Table 5.3 identifies the average number of optimal solutions that were present between the source and destination vertices on the selected graphs. For each vertex pairing the set $PfTRUE$ was retrieved and the average value for $|PfTRUE|$ generated. The second of the three columns identifies the K number of iterations required to generate the set $PfTRUE$. The final column provides the runtime in milliseconds to generate the set $PfTRUE$, or alternatively expressed to perform a

single run of Dijkstra shortest path algorithm on a criterion then produce the K paths required on the alternative criteria and finally extract the optimal set of solutions from the set (of Dijkstra and K paths) generated. The results indicate that in certain circumstances, notable with a small delta value, the algorithm will only be required to generate a limited number of K paths and will therefore be able to complete the process of *PfTRUE* assembly reasonably quickly.

In Table 5.4 presents the results of a similar test where the difference between the shortest paths in terms of distance and time are larger, as opposed to smaller. The presence of larger delta values would indicate that a high number of paths will exist between two nodes and will therefore result in a higher number of K paths required (indicated in the second of four columns) and will result in a longer runtime. A time limit of 5 minutes was applied to the test as a terminating condition. The table provides the same information as Table 5.3 in the first three of the four columns, with the addition of the success rate of the algorithm in the fourth and final column. Only those graphs where more than 60% of the vertex pairings were acquired within the time limit are given. The experimentation using the Climaco and Martins algorithm is concluded with Table 5.5. The table includes the average K value for each vertex pairing together with standard deviation for the K value. It should be noted that the five minute terminating criteria was not chosen at random. During tests it became apparent that where the test could not be completed in less than five minutes in over 80% of cases the system would fail to complete the process and report that the amount of memory required exceeded that available. That increased to over 90% in cases where the terminating condition was set to 10 minutes. Figure 5.4 provides a visual analysis of the number of paths required while Figure 5.5 compares the runtimes seen.

Graph	Average <i>PfTRUE</i>	Average K	Average Time (ms)
Cardiff 250	3.3	14	20
Cardiff 500	4.1	48	198
Cardiff 750	4.3	54	212
Cardiff 1000	4.5	65	324
Cardiff 1500	4.3	73	690
Cardiff 2000	6.3	143	963
Cardiff 4000	8.2	160	10836
London 250	3.8	34	34
London 500	4.1	41	310
London 1000	4.3	156	1294
London 2000	7.8	415	4654
NE 1000	2.08	2.26	86
NE 2000	2.6	3.12	94

Table 5.3 Performance of Climaco and Martins Algorithm with Low Delta Values

Graph	Average <i>PfTRUE</i>	Average K Calculated	Average Time (ms)	Successfully Completed (%)
Cardiff250	3.3	18	22	100
Cardiff500	6.1	2276	11237	92
Cardiff750	7.2	3412	12072	81
London250	5.6	1056	2920	100
London500	12.5	12564	33937	71
NE1000	3.78	25	1481	100
NE2000	3.78	26	2791	100

Table 5.4 Performance of Climaco and Martins Algorithm with High Delta Values

Graph	Low Delta Spacing		High Delta Spacing	
	Average K Value	Standard Deviation in K	Average K Value	Standard Deviation in K
Cardiff 250	14	4	18	4
Cardiff 500	48	12	2276	515
Cardiff 750	54	14	3412	812
London 250	34	8	1056	768
London 500	41	13	12564	8634
NE 1000	2.26	1	25	1
NE 2000	3.12	1	26	3

Table 5.5 Comparison of Results Seen in Climaco and Martins Approach

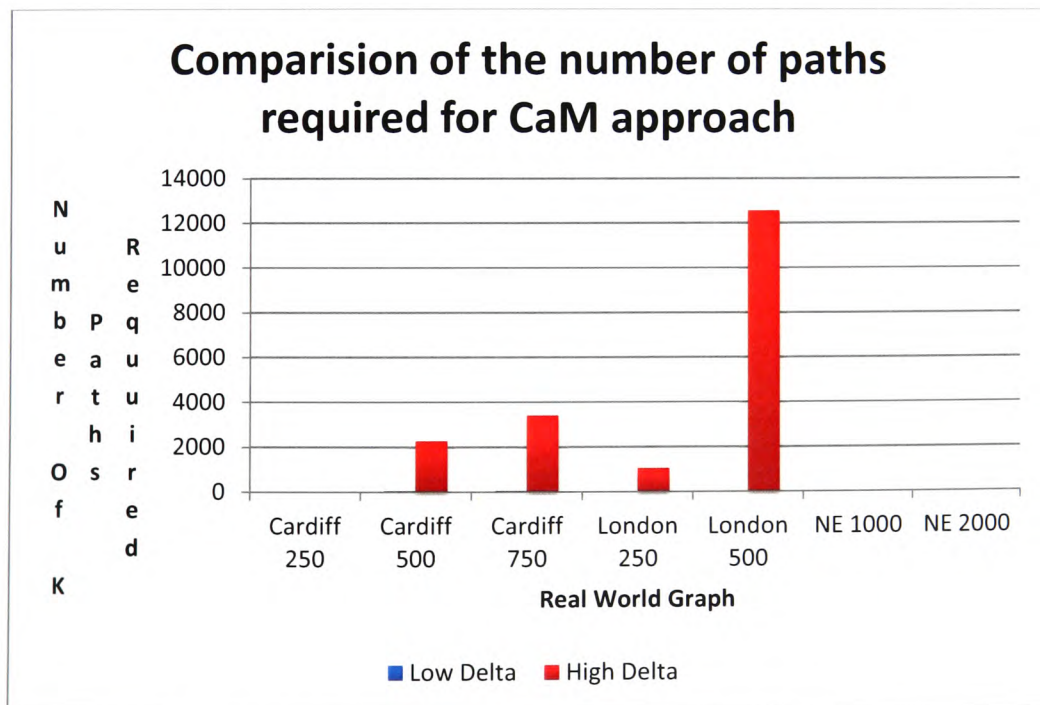


Figure 5.4 Comparison of the Paths Required for C&M Approach

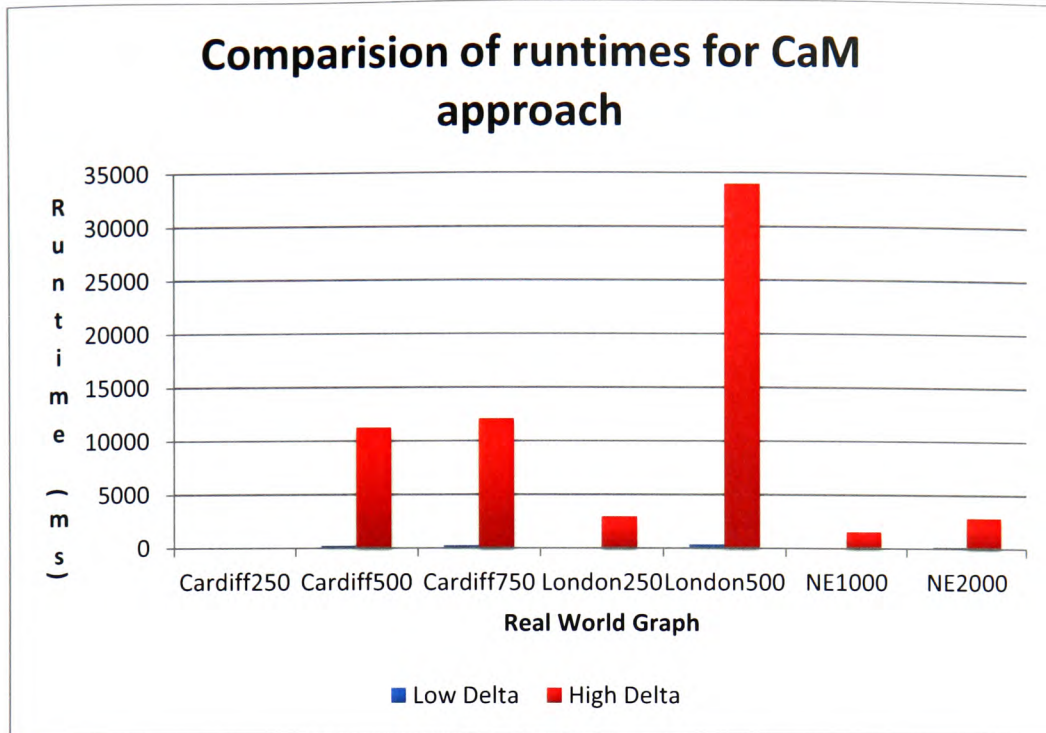


Figure 5.5 Comparison of the Runtime Required for C&M Approach

5.1.4. Observation Regarding the Algorithmic Approach

In this section the three algorithms for the solution of the MSPP are compared. As indicated previously the application of the Dijkstra algorithm to multiple criteria produces only the extreme points of the front $PfTRUE$. The Skriver and Anderson algorithm has been applied to both a series of real world and random graphs. The results obtained for that experiment indicate a substantial increase in processing time as the size of the graph increases. On medium sized real world graphs (Cardiff 1500) the algorithm took almost eleven minutes to process a solution. Due to the long run times offered by that algorithm on smaller graphs it was decided not to proceed on the larger graphs (considering that the Cardiff 1500 graph has the dimensions 6603 x 14428 while the Cardiff 4000 graph has the dimensions 29044 x 62486). Under certain circumstances the Climaco and Martins approach has the ability to substantially outperform that of Skriver and Anderson. Consider that on the Cardiff 500 graph the Skriver and Anderson algorithm will take around 101 seconds compared to 11 seconds

for the Climaco and Martins approach in certain circumstances. The Skriver and Anderson approach is however able to complete runs regardless of the vertex selection a property not seen in the Climaco and Martins approach. The Dijkstra shortest path algorithm is able to return an extreme sub subset of $PfTRUE$ very efficiently on real world graphs.

The tests on the Climaco and Martins approach show that the performance of the algorithm is entirely dependent on vertex pair selection. In some circumstances the methodology performs well with the algorithm quickly developing an accurate and complete set of $PfAPPROX$. In other situations the algorithm performs poorly with the methodology demonstrating poor run times. Without prior knowledge of both the graph and the vertex pairing there is no opportunity to detect such scenarios where the algorithm will perform poorly.

5.2. Analysis of Runtimes of Heuristic Approaches

In the experiments upon each of the algorithms it was originally the aim to develop parameter sets that would enable comparable runtimes across algorithms (excluding the K-Geodesic approach to the Genetic Algorithm) and graph sizes. It quickly became apparent however that such an approach would prove impossible. A rational for the discrepancy is detailed in section 5.3 . In this section the runtimes of the algorithms are discussed. The time taken for the algorithms to perform the analysis on a subset of the graphs studied using the smallest of the parameter sets are captured. The results acquired using random graphs are provided in Figure 5.6 Comparable differences in run time performance can be seen across the parameter size sets. The parameters used are formally introduced in Chapter 6.

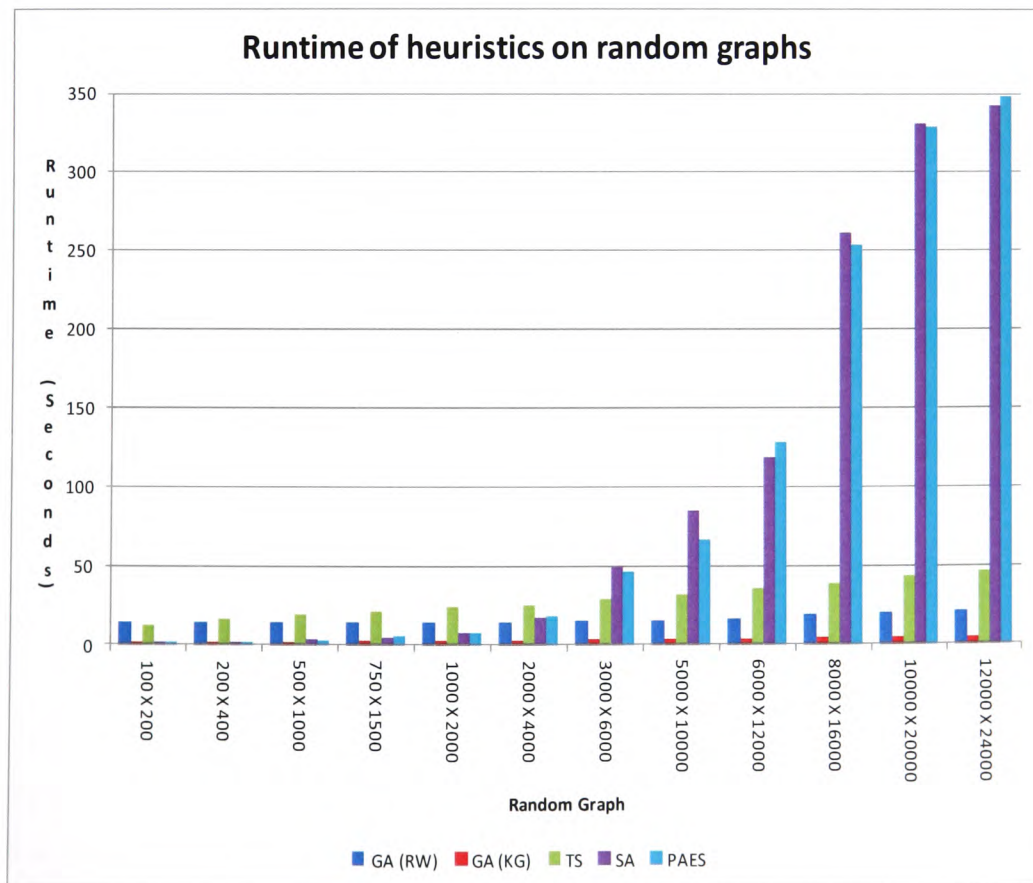


Figure 5.6 Runtime of Heuristics on Random Graphs

The K-geodesic approach completes a processing operation extremely quickly, completing in 5.1 seconds on the largest of the graphs. The technique typically takes around one quarter of the time the random walk directed technique on the same graph and 1.5% of the time of the Simulated Annealing approach on graph size 12000 x 24000. The comparative uniformity of the GA (both variants) approach when compared to the 1+1 series of algorithms is noted and discussed further in section 5.3.

It is perhaps worth highlighting a certain discrepancy between the runtime of the K shortest path algorithm when producing geodesic paths and the results seen in certain instances of the Climaco and Martins algorithmic approach. During the experiments performed on the Climaco and Martins approach, the algorithm is required to perform several dictionary lookups to retrieve the cost of a path. With the K-geodesic approach the cost value is fixed to one resulting in a reduced computational overhead. In addition although the output of the K shortest path algorithm may be seen as a simple path internal to the algorithm complex paths may be generated and then discarded. The application of the K-geodesic approach sees less generation of complex paths and hence a faster runtime. A test setting K to 1000 on a graph size 5000 x 15000 was performed using both the K Geodesic approach (edge cost set to 1) together with a single criteria cost associated with the graph. The K Geodesic approach internally required the computation of an average of 1006 paths compared with an average 1401 paths generated using the single criteria cost. Runtimes of 1162 milliseconds compared with 1621 milliseconds using the K geodesic and single criteria edge cost respectively. The test was performed over 1000 randomly selected vertex pairings.

Figure 5.7 presents the run times achieved by the various algorithms on real world graphs. On the larger instances of the real world graph the Simulated Annealing and PAES algorithms failed to complete within two hours. For the Cardiff 2000 graph the random walk based Genetic Algorithm provides a solution to the MSPP task in around 24 minutes. The K geodesic approach performs much more quickly completing with a maximum runtime of 26.1 seconds; the Tabu Search completes in around 28 minutes.

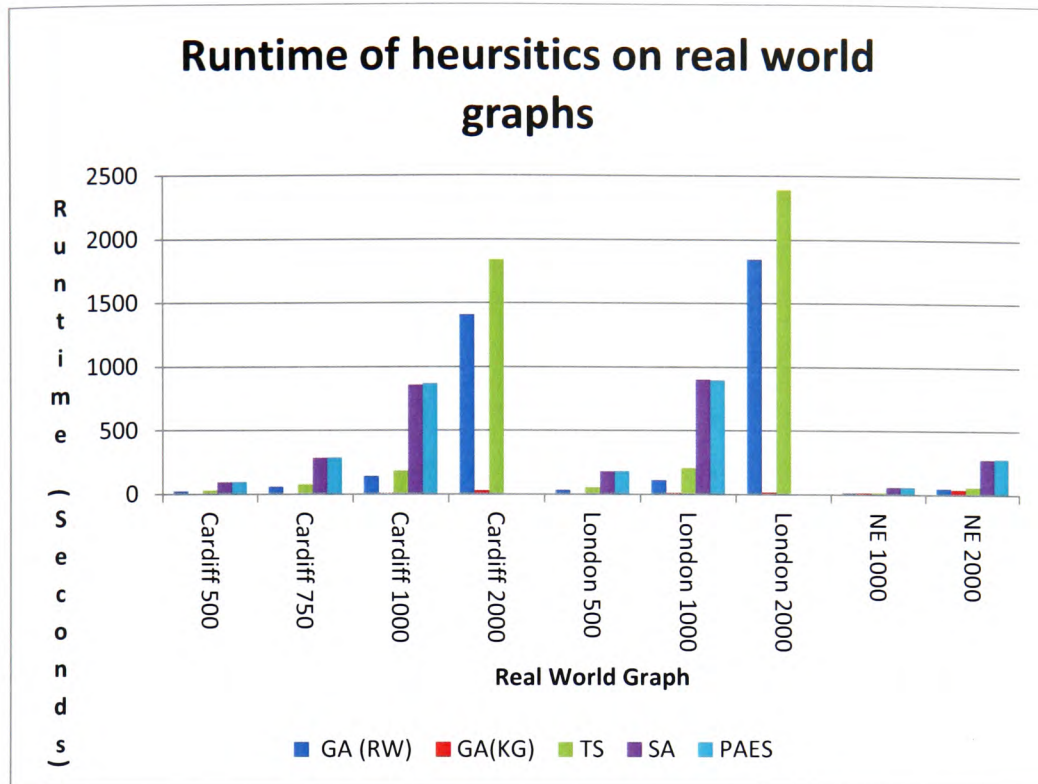


Figure 5.7 Runtime of Heuristics on Real World Graphs

The heuristic algorithms (with the exception of the K Geodesic approach) may appear to be disappointing. However when compared to existing methods the algorithms are seen to produce a reasonable approximation of $PfTRUE$ in a useable period. The Cardiff 1000 graph for instance using the Genetic Algorithm with random walking produces a useable solution in 139.63 seconds compared to 331 seconds for the Skriver and Anderson approach. Where tree based approaches are compared the K-geodesic performs very well (in terms of runtime) compared to the Climaco and Martins approach. The approach is able to produce a reasonable approximation of the front $PfTRUE$ on the Cardiff 1000 graph in 6.9 seconds where the Climaco and Martins approach fails to return a solution within five minutes (and would likely cause system errors).

The results of runtimes gathered from the test performed on random graphs are certainly promising and exhibit a degree of comparability as the graph size increases. This comparability is not seen in existing work. Mooney (2004) reports a runtime of around 114 seconds on a graph sized (1000 x 2000) for a Genetic Algorithm approach. Using similar parameter values the methodology here performs much quicker (14 seconds). The difference is believed to be due to the difference in random walk techniques employed in either study. In virtually all other aspects the algorithms are the same with our algorithm based upon that of Mooney (2004). A caveat is attached to this assertion however. The same increase in performance does not *appear* to be present on real world graph, at least with the same degree of comparability. However, due to differences in data sets an exact comparison is not possible.

The runtimes seen on the real world graphs when compared to their synthetic counterparts demonstrate how the makeup of the graph plays a vital role in the performance of the algorithms. The real world graphs exhibit a much higher degree of sparseness than the random graphs. The graph will tend to have a large number of vertices with only two connected edges one of which will already be present in the walk. The random walk however will still need to identify the number of outgoing vertices thus decreasing the performance of the mechanism.

5.3. Reviewing the Runtime Discrepancy

In section 5.2 the runtime of the various algorithms were considered. In that section it was highlighted the original aim of the work was to present the algorithms that run in a comparably similar time. It quickly became apparent that such an approach would prove impossible. In this section two of the heuristic approaches, the Genetic Algorithm (with random walking) approach and the PAES are used to provide a rationale for the discrepancy. Figure 5.6 contains the run times of each of the algorithms on a subset of the randomly generated graph. The fact that the Genetic

Algorithm approach maintains a comparatively similar operating time across the random graphs should be highlighted. In this section, a rationale for the comparability across random graph sizes for the Genetic Algorithm is proposed.

The Genetic Algorithm requires that at each generation (G) the population (P) only consist of unique paths. In this thesis the same path is allowed to be readmitted at a later generation but for any specific generation each path must be unique. On a smaller graph there are likely to be less unique paths seen on the graph between any pair of vertices. As the size of the graph increases the number of feasible paths between any two vertices also increases reducing the number of attempts that are required to discover a complete set of unique paths. Table 5.6 contains the total number of calls required to the random walking mechanism when performing a run of the Genetic Algorithm on graphs at either end of the graph size extremes. For the smaller graphs each walk is performed exceptionally quickly, completing in around 0.06 milliseconds. However a large number of calls to the random walk mechanism are required in order to produce enough random walks to generate a series of unique populations. In contrast on the larger graph size less duplicate paths are found per generation but each of which (a path) takes longer to be generated taking around 5-8 milliseconds per walk.

Graph Size	Total Paths Generated
100 x 200	246,216
12000 x 24000	3911

Table 5.6 Total Paths Generated Per Run of Genetic Algorithm

The general effect is to balance out the run time of the Genetic Algorithm across the series of random graphs. The PAES algorithm together with the other heuristic algorithms in comparison will require less walks (iterations). This on the small graphs

results in a quicker run time. However, the same number of walks are required on the larger graph sizes which results in an increasing run time due to the greater amount of time required for their generation. The review of runtimes is completed by a brief description of the admission of unique paths. Figure 6.3 demonstrates that the Genetic Algorithm will admit an average of 781 unique paths on the graph size 100 x 200. The PAES running on the same graph admits the lower number of 719 unique paths. On larger graphs (12000 x 24000) the PAES algorithm admits a much higher number (1365) of unique paths than the Genetic Algorithm (846), but the general difference in quality remains. This indicates that on larger graphs the performance of the algorithm is heavily influenced by the genetic operators of crossover and mutation.

5.4. Scalability of Criteria

In the next brief experiment the ability of the algorithms to scale to higher numbers of optimisation criteria is examined. The tests conducted so far have been limited in the number of criteria under consideration and consist of either two or three criteria, primarily two. The multi objective approach to the Dijkstra algorithm in which four criteria were considered is not included as tests using those datasets on the heuristic approaches were not considered. Figure 5.8 presents the runtimes achieved with higher numbers of criteria. For the purposes of brevity the discussion is limited to the results achieved from the Genetic Algorithm using random walking using parameter Set A, described in Chapter 6 (6.2.1). However, the logic of calculating and extracting the Pareto optimal path is relatively straightforward and comparable results can be seen across the range of algorithms considered. In the previous section a discussion as to the makeup of the algorithms runtimes was provided. The algorithms demonstrate a high degree of scalability with little difference being observed regardless of the number of criteria considered. It should be noted that in this particular experiment the sole test was to examine the performance in terms of runtime of the algorithms and not the quality of the solutions returned. Four graphs sizes stand out as being worthy of further analysis and discussion - the graphs 1000 X 2000, 2000 X 4000, 6000 X 2000 and finally 12000 X 24000. The first of the four graphs exhibit very little variation in runtime regardless

of the number of criteria (less than 1 hundredth of a second). The largest graph has a high difference between $C=3$, $C=5$ and $C=10$. Further analysis results in Table 5.7 that provides the average runtime seen for each graph across all criteria together with the difference from that average seen for each criteria group. The final column presents the typical difference from the average runtime for that graph size. Reviewing the graph sized 12000 X 24000 on Figure 5.8 reveals a wide range of runtimes across the number of criteria being considered with almost four seconds difference between $C=3$ and $C=5$ on the largest graph. For the largest of the graph the typical difference from the average runtime seen is 1.18 seconds. The difference in runtime between $C=3$, $C=5$ and $C=10$ on the largest graph is around $\pm 8\%$ of the runtime of the run where $C=5$. The difference is high but importantly the difference does not increase in line with the number of criteria being considered. This is consistent across all graphs. For instance on the 100 X 200 the average runtime returned from $C=10$ is quicker than $C=5$. On the graph 200 X 300 $C=3$ performs more quickly than both $C=2$ and $C=5$ as does $C=10$. An exact rational for the run time difference cannot be identified but could be either internal to the algorithms such as random walk finding a large number of identical paths or external and caused by the operating system background process.

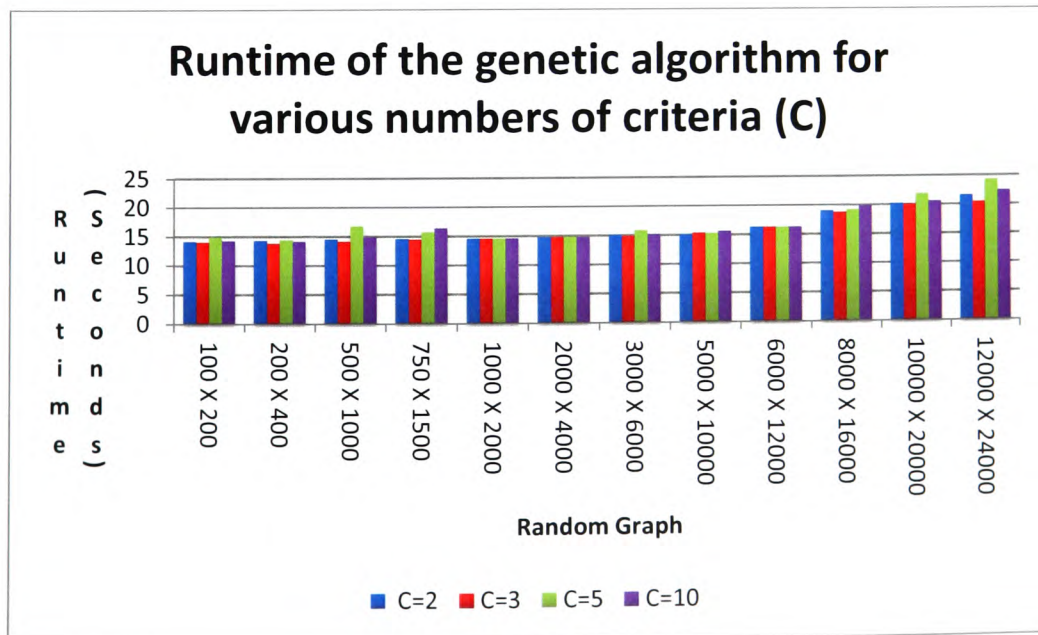


Figure 5.8 Runtime of the Genetic Algorithm for Various Numbers of Criteria

		Difference From Average Runtime (Seconds)				
	Average Runtime (Seconds)	C=2	C=3	C=5	C=10	Average Difference (Seconds)
100 X 200	14.39	0.17	0.29	0.53	0.07	0.265
200 X 400	14.2275	0.1225	0.3375	0.2325	0.0175	0.1775
500 X 1000	15.14	0.55	0.95	1.68	0.18	0.84
750 X 1500	15.3525	0.6925	0.8125	0.4275	1.0775	0.7525
1000 X 2000	14.67	0	0	0	0	0
2000 X 4000	14.8	0	0	0	0	0
3000 X 6000	15.2925	0.1725	0.3725	0.5975	0.0525	0.29875
5000 X 10000	15.3925	0.2425	0.0175	0.0525	0.2775	0.1475
6000 X 12000	16.35	0	0	0	0	0
8000 X 16000	19.2525	0.1525	0.4625	0.0225	0.6375	0.31875
10000 X 20000	20.7425	0.4925	0.5925	1.2075	0.1225	0.60375
12000 X 24000	22.2	0.6	1.75	2.1	0.25	1.175

Table 5.7 Analysis of Runtime Differentials

5.5. Summary of Quality Tests

Figure 5.9, Figure 5.10 and Figure 5.11 provide a summary of the results seen during the experimental work conducted as part of this research. Figure 5.9 highlights for each of the algorithms studied as part the work the number of tests where the size of the approximation front is equal to or greater than that of the set of pre-determined optimal solution, or $|Pf_{APPROX}| \geq |Pf_{TRUE}|$. Figure 5.10 highlights the number of tests where the difference between the two sets is equal to one ($|Pf_{APPROX}| = |Pf_{TRUE}| - 1$), while Figure 5.11 presents the number of tests where the difference the Pf_{APPROX} and Pf_{TRUE} is equal to or greater than 2 i.e. the approximation of the optimal solutions has two or more solutions missing or $|Pf_{APPROX}| \leq |Pf_{APPROX}| - 2$.

2]. The parameters selected for the experiments are discussed in detail in Chapter 6, along with a series of experiments considering multiple parameter values.

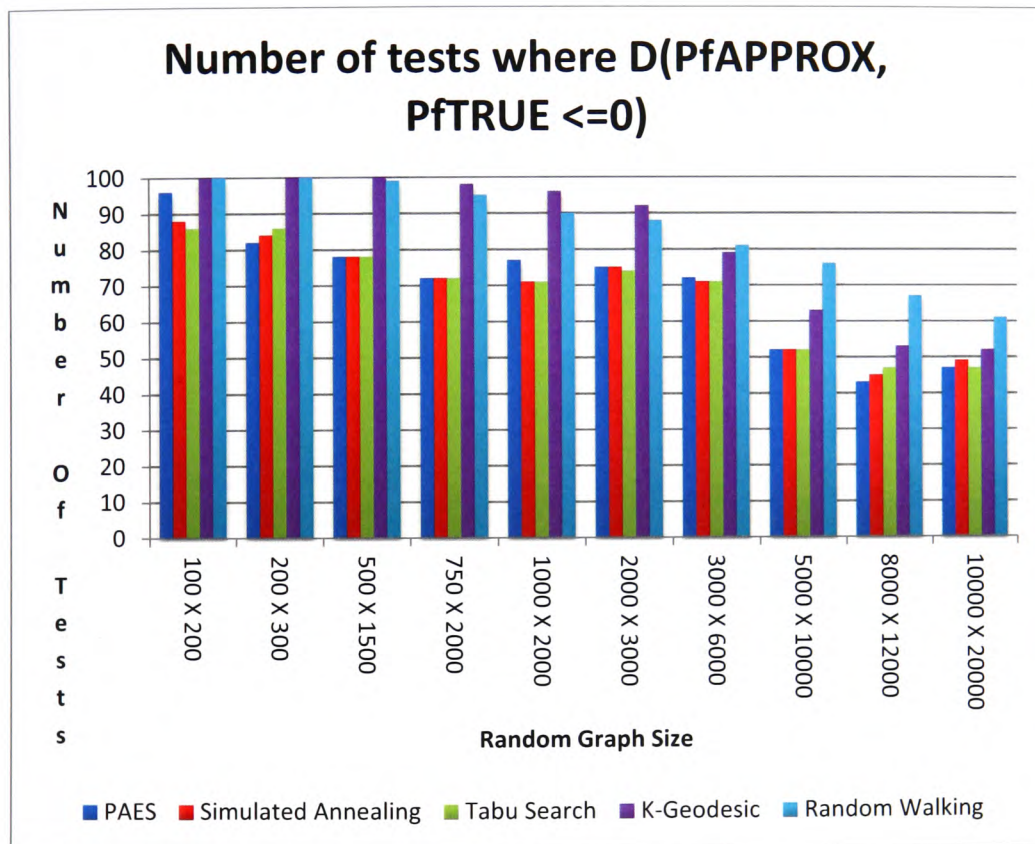


Figure 5.9 Tests Where $D(\text{PfAPPROX}, \text{PfTRUE}) \leq 0$

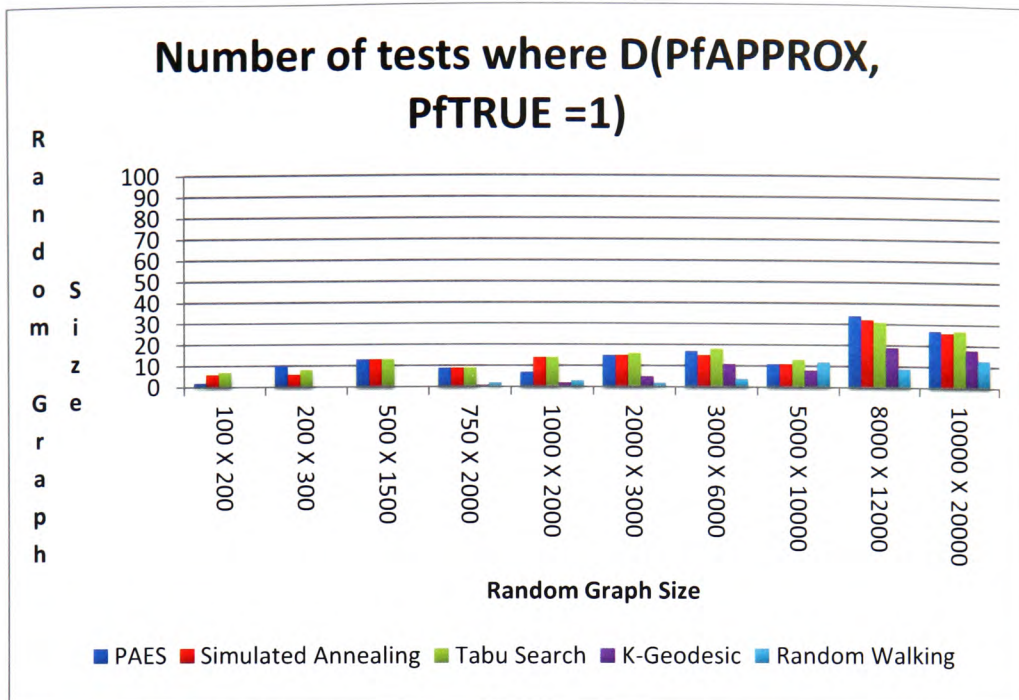


Figure 5.10 Tests Where $D(\text{PfAPPROX}, \text{PfTRUE}) = 1$

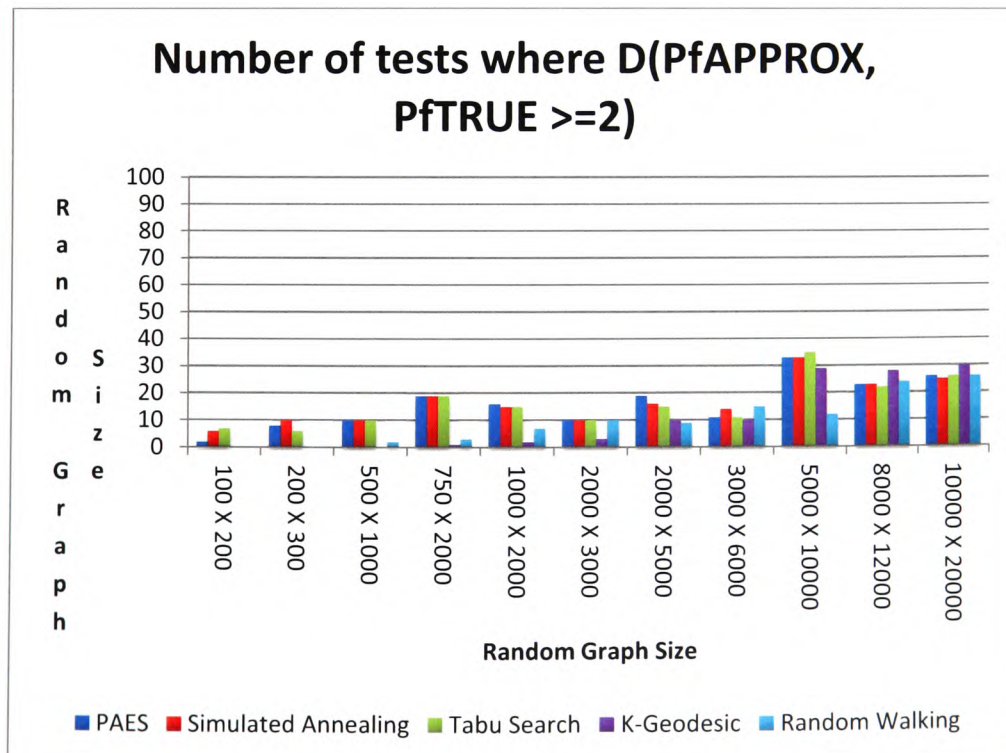


Figure 5.11 Tests Where $D(\text{PfAPPROX}, \text{PfTRUE}) \geq 2$

On smaller sized graphs the Genetic Algorithm approach is able to generate an approximation set equal to or larger than set of optimal solutions in the majority of cases. The Tabu search demonstrates a lower level of performance on those same graphs but is still able to return over a high quality approximation in over 85% of the tests performed. Including the tests where only a single solution is absent from the metrics increases the levels success seen to over 90% for each of the heuristics approaches. Increasing the size of the graph has a noticeable effect on the quality of the results for all the algorithms, although this is a reasonable behaviour not at all unexpected. The Genetic Algorithms are able to return a complete approximation in over 50% of cases on the largest of the graphs and are the only heuristic approaches that are able to offer this level of performance. The other heuristic approaches are able to offer the same levels of performance in around 46-48% of cases. The alternative approaches to the Genetic Algorithm however are often only missing a single solution from the approximation sets generated by each approach. Figure 5.12 presents the results where the both Figure 5.9 and Figure 5.10 are combined ($|Pf_{APPROX}| \geq |Pf_{TRUE-I}|$) into a additional set considered as being of high quality. As seen in Figure 5.12, other than in isolated cases such those seen on the graph sized 5000 X 10000 where the Genetic Algorithm has a spike in the number of high quality solutions the algorithms are generally comparable. Excluding the isolated spike seen on the graph 5000 X 10000 using the genetic algorithm the typical spread across all algorithms and graph sizes is 9 with a maximum of 18 (750 X 2000) and minimum of 5 (graphs 3000 X 6000 and 10000 X 20000). Chapter 6 provides an in-depth analysis of each of the heuristic approaches together with a description of the behaviour whilst performing the analysis.

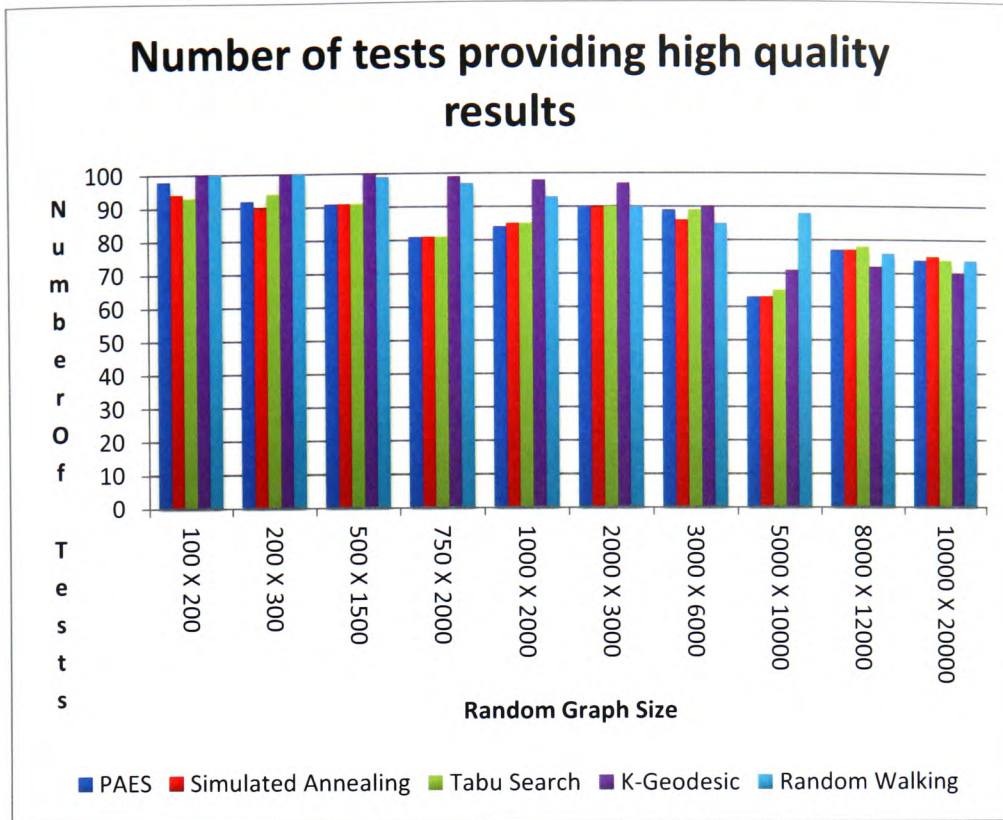


Figure 5.12 Tests Producing High Quality Results

5.6. Admission of Locally Optimal Paths

In this section, the admission of locally optimal paths into the front $PfAPPROX$ is measured. The admission of such solutions would, from a purely theoretical standpoint indicate poor performance for each of the algorithms. However, when considering the MSPP and where an assessment is based upon how people select a route from the options provided the presence of such solutions does not necessarily indicate such a poor performance.

Figure 5.13 presents the number of instances where locally optimal paths are seen in those solutions where $|PfTRUE| \leq |PfAPPROX|$. For each of the algorithms the number of cases where locally optimal solutions are seen is provided. Appendix B contains complete result information in tabular form.

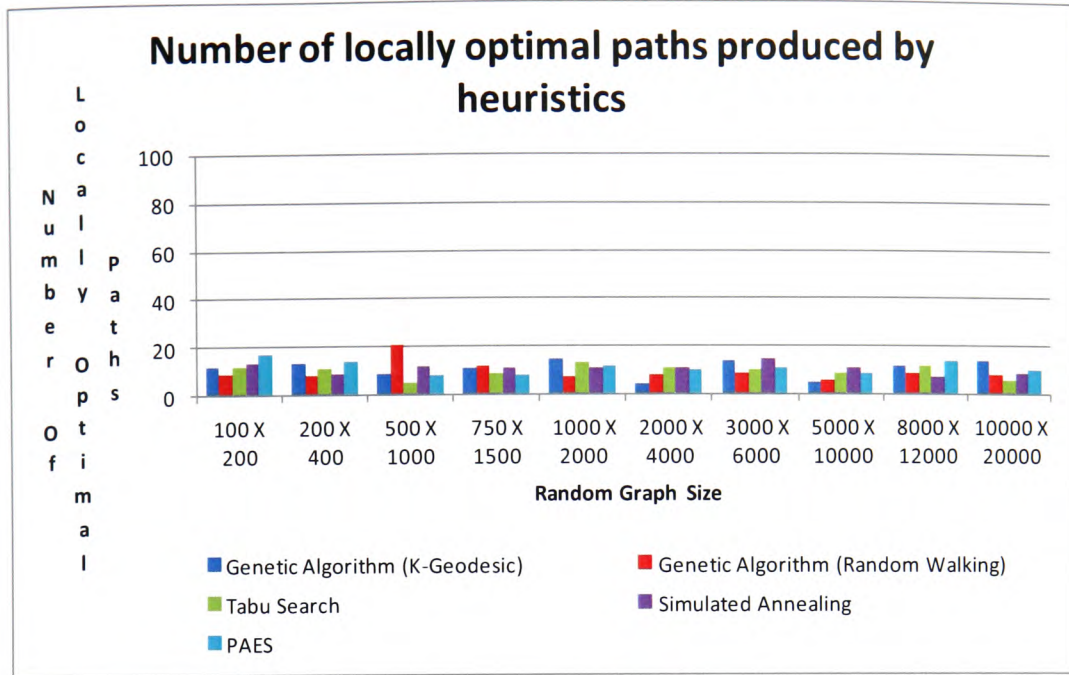


Figure 5.13 Number of Locally Optimal Solutions Produced by Heuristics

For each of the solutions where $|PfTRUE| \leq |PfAPPROX|$ the distance between each solution admitted into $PfAPPROX$ and the closest corresponding neighbour in the front $PfTRUE$ is measured. Table 5.8 presents a view of what *may* be considered high, medium and low quality results based on the bounds selected during the creation of the graphs for this study. The metrics are qualitative based upon the visual inspection of a large proportion of the fronts admitted. Figure 5.14 highlights the average distance seen where values close to zero would indicate a high quality approximation of the front. The distance metrics are domain specific and based upon the upper and lower bounds of the graph specified during generation. Attention is drawn to Figure 6.19 in the following chapter which provides an example of the scenario where six members are present in $PfTRUE$ but the Tabu Search returns seven. A visual examination however highlights a good overall approximation with an average distance value of 2.25. Scenarios such as $|PfAPPROX| < |PfTRUE|$ are not covered in the test.

Although the quality tests concentrate on random graphs similar results are seen on real world graphs. Figure 5.15 highlights an example of a real world scenario where the front $PfTRUE$ is larger than that seen in the front $PfAPPROX$. Visual examination of Figure 5.15 shows that the approximated front contains two members of $PfTRUE$. There are three locally optimal paths discovered which are very close to two members of $PfTRUE$. Using real world metrics the approximations are within 2-3 meters of the length of the optimal paths with a travel time difference of around 2-4 seconds. The example is derived from the graph Cardiff 1000 using parameter set E of the Genetic Algorithm. The overall distance metric is 3.16 yielding a good approximation of the front. Exact details of the parameters used are provided in Chapter 6.

Quality	Minimum	Maximum
High	0	3.49
Medium	3.5	5.99
Low	6	

Table 5.8 Comparative Qualities of Distance Metrics

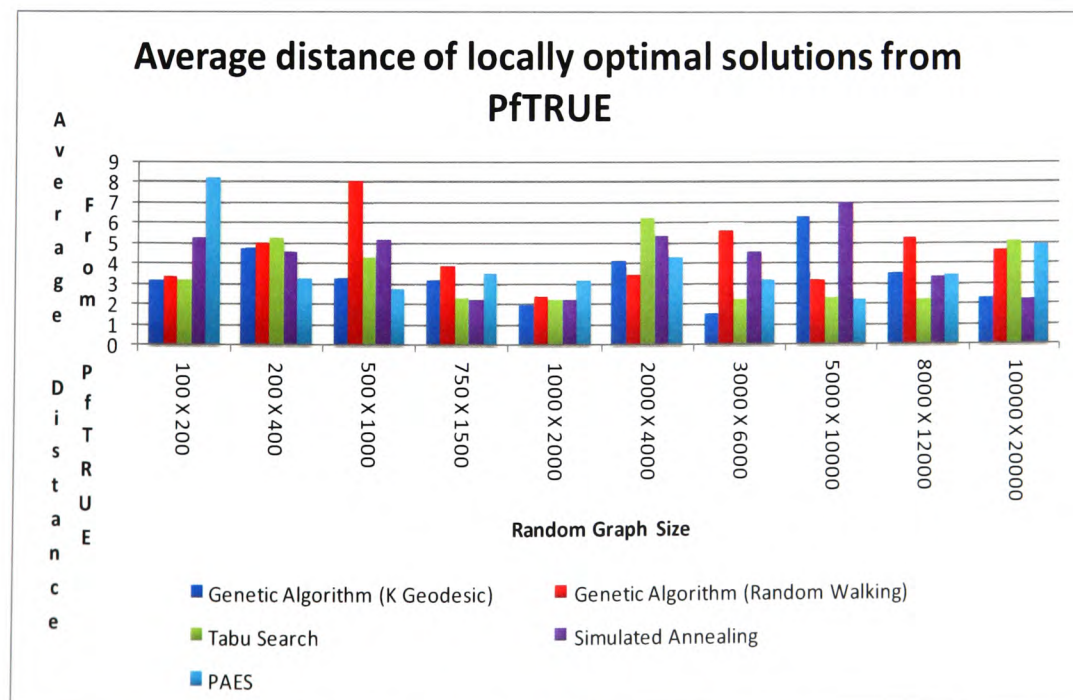


Figure 5.14 Average Distance of Locally Optimal Solutions to $PfTRUE$

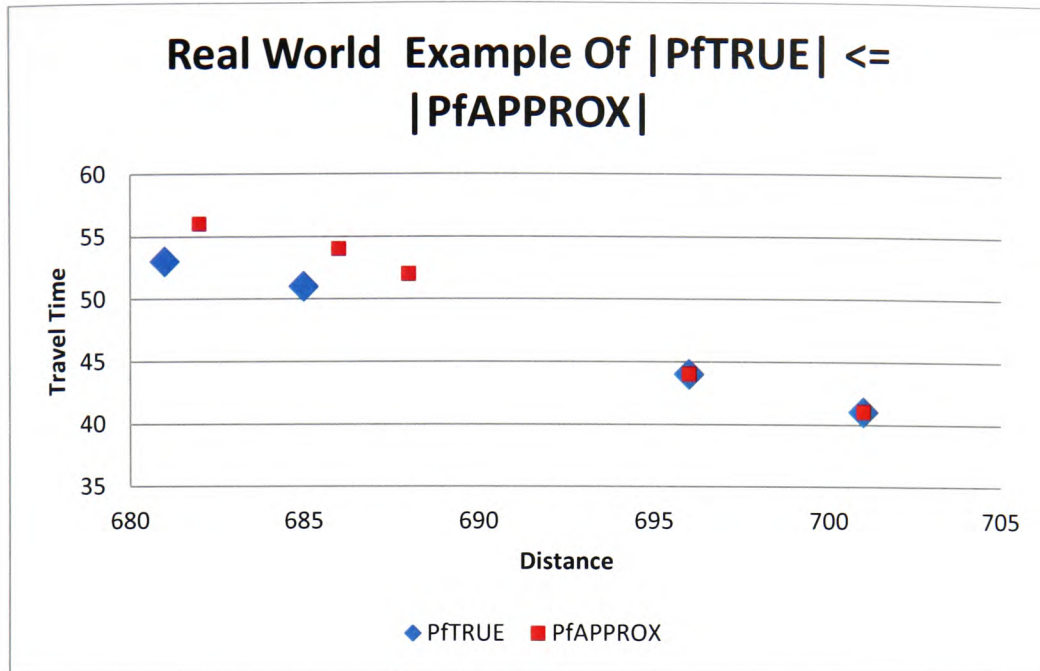


Figure 5.15 Real World Example of $|PfTRUE| \leq |PfAPPROX|$

5.7. Cross Algorithm Analysis

The aim of the current section is to provide a higher-level analysis into the appropriate selection of the various approaches for solving the MSPP depending on factors such as the time available to perform the analysis or the quality required from the analysis. It is somewhat ironic that the process of experimentation undertaken has failed to identify a single optimal approach for the solution of the MSPP. That is to say there is no universal algorithm or methodology best suited to the solution of the MSPP across all graph sizes. The lack of optimality for any single method extends beyond the use of the various heuristic approaches and is inclusive of the non heuristic methods studied as part of this work. Here an attempt is made to provide a general synopsis into what may be considered the appropriate algorithm dependant on two factors, firstly the amount of time available to perform the analysis and secondly the quality of the output required from that same analysis. Figure 5.16 is introduced in an attempt to highlight the disparate nature to the methods used in the solution of the MSPP considered for this work. The results obtained from the experimental phase of the work

indicate highlight it is possible to obtain a complete and accurate set of optimal solutions if the algorithms are run until completion. Conversely, the extreme points of the Pareto optimal front can be acquired very quickly but may be considered a simplistic approximation of the optimal front.

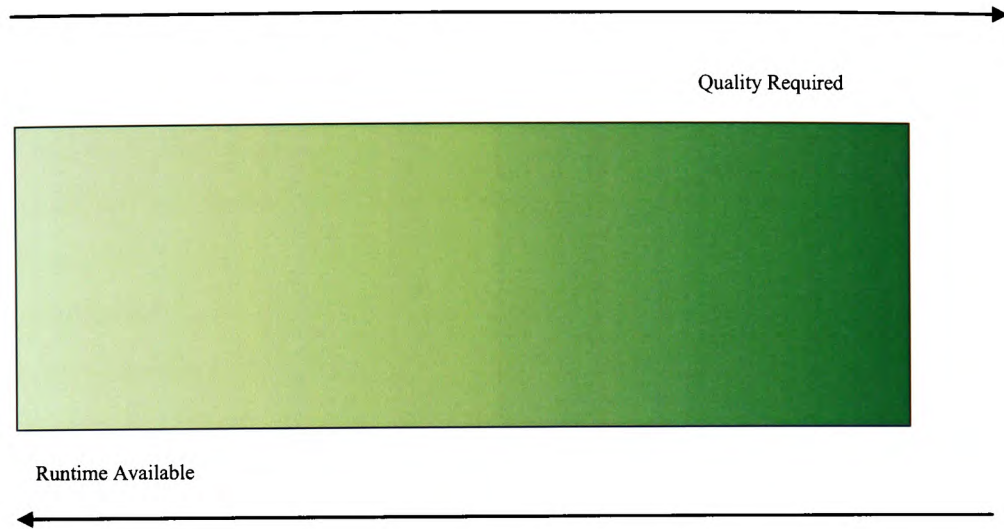


Figure 5.16 Diverging Properties of MSPP Approaches

In order to demonstrate the performance of the algorithms Figure 5.17, Figure 5.18 and Figure 5.19 are introduced. Figure 5.17 presents the runtimes of a series of approaches undertaken in this work on a graph sized 100 X 200 while Figure 5.18 presents the runtimes seen on a graph size 1000 X 2000. Each gives the runtimes of the Dijkstra shortest path algorithm run separately for each criteria, the runtimes for each of the four heuristic techniques considered for this work (the Genetic Algorithm, the Tabu Search, Simulated Annealing and PAES). Finally, we present the runtimes of the Skriver and Andersen (SaA) algorithmic approach. Figure 5.19 is introduced which presents the runtimes on a graph sized 10000 X 20000. In terms of runtime the Skriver and Andersen approach often took over 30 minutes and is not included in Figure 5.19 for clarity purposes. The inclusion of that method overpowers the remaining methods reducing the value of Figure 5.19. It should however be accepted that the method does return the complete optimal set of solutions.

In terms of the methods considered the Climaco and Martins (CaM) algorithmic approach might be considered notable for its absence in Figure 5.17, Figure 5.18 and Figure 5.19 given that earlier in this work it has been highlighted that the in certain circumstances that approach has been shown to outperform that of SaA. The reason for its absence is largely due to the fact that without prior analysis of the graph structure it is not possible to determine if the CaM method will outperform that of SaA. In addition the CaM algorithm outperforming the SaA approach occurs on only certain graph types and is not universal. For those reasons the CaM method is not considered here although attention is drawn to the fact that the approach *may* on real world graph examples outperform the SaA method. Algorithm 5.1 is introduced however which attempts to run both algorithms until either has been completed, thus enabling possible faster analysis to be completed. Due to the differential relating to the time scales between the graph sizes and algorithms the data for Figure 5.17, Figure 5.18 and Figure 5.19 is included in Table 5.9.

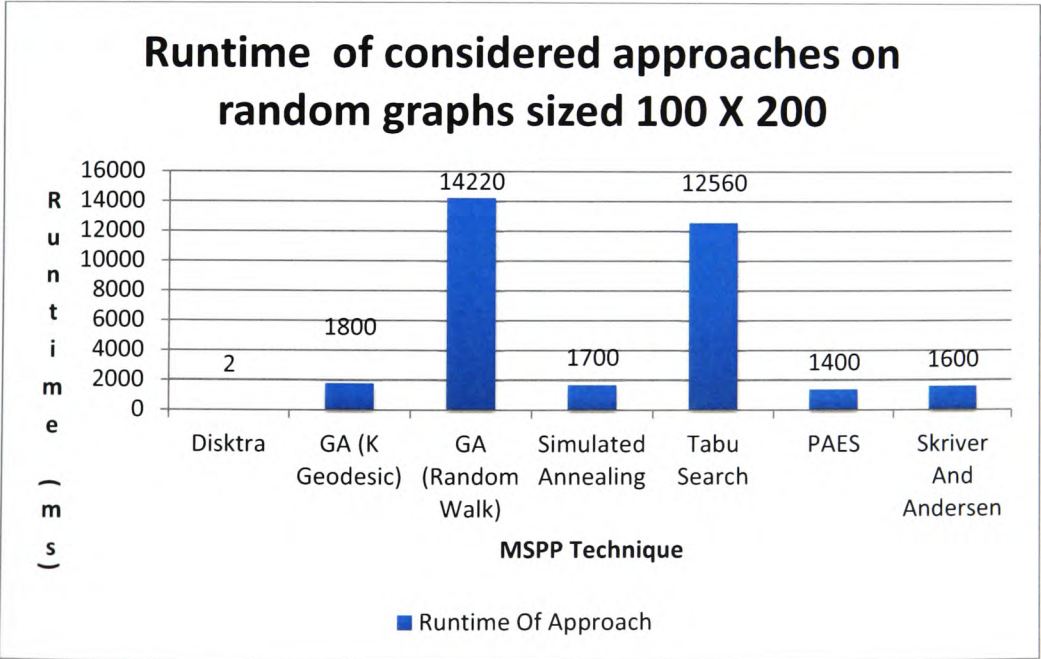


Figure 5.17 Runtime of Approaches on Small Sized Graphs

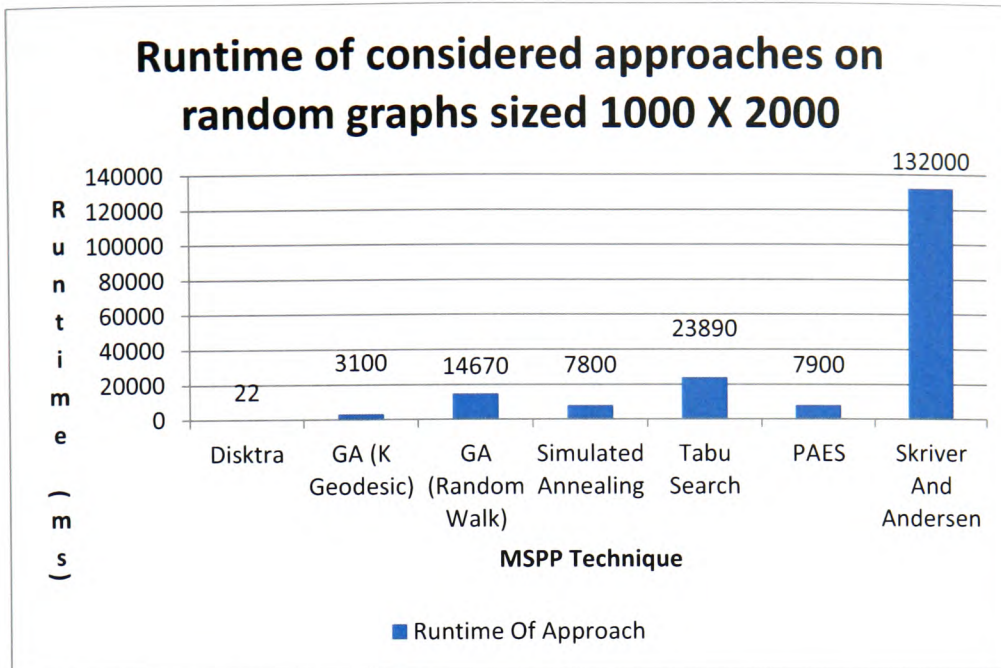


Figure 5.18 Runtime of approaches on a Medium sized graph

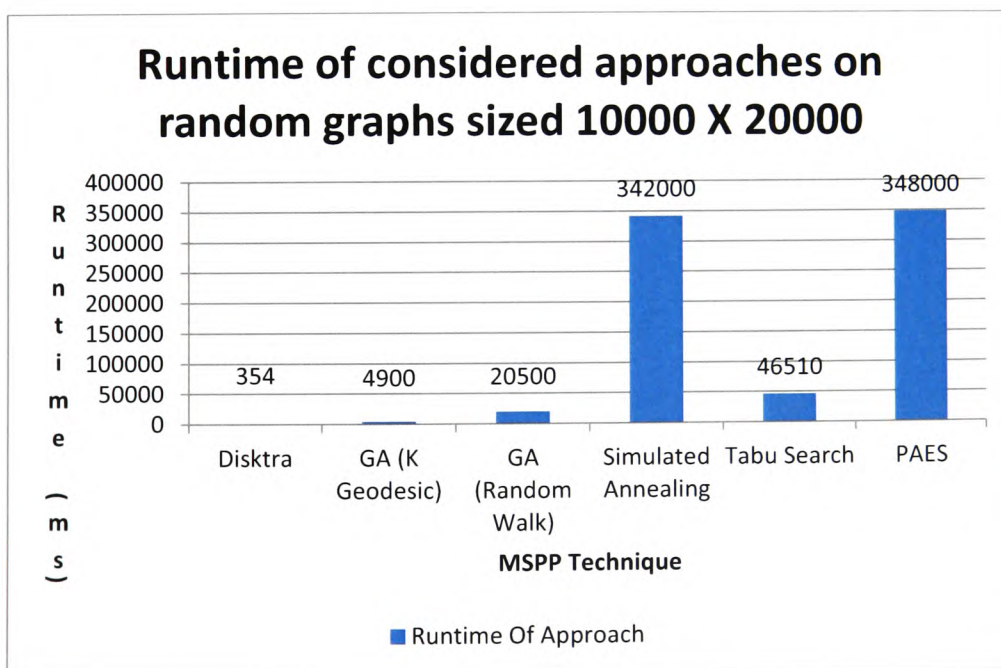


Figure 5.19 Runtime of Approaches on a Large Graph

Runtime (ms) Of Various MSPP Techniques On Random Graphs							
Graph	Dijkstra (Multi Objectives)	GA (K Geodesic)	GA (Random Walk)	Simulated Annealing	Tabu Search	PAES	Skriver And Andersen
100x200	2	1800	14220	1700	12560	1400	1600
1000x 2000	22	3100	14670	7800	23890	7900	132000
10000 x 20000	354	4900	20500	342000	46510	348000	

Table 5.9 Runtime of Various MSPP Techniques on Random Graphs

Figure 5.17, Figure 5.18 and Figure 5.19 show that the performance of the Dijkstra shortest path algorithm requires very little computational effort in comparison to the other techniques reviewed. On the largest of the graphs as shown in Table 5.9 the technique requires just 354 milliseconds despite handling additional criteria in those tests where four criteria are considered rather than two when considering the heuristics and the SaA approach. Taking the runtimes of the Dijkstra algorithm with multiple runs into consideration then it is possible to populate to regenerate Figure 5.16 to take into account the performance offered by the Dijkstra shortest path algorithm. Where computational performance is required rather than a high quality approximation of the optimal set the application of the Dijkstra shortest path algorithm is appropriate. Figure 5.20 updates Figure 5.16 to take into account the appropriate selection of the Dijkstra shortest path algorithm.

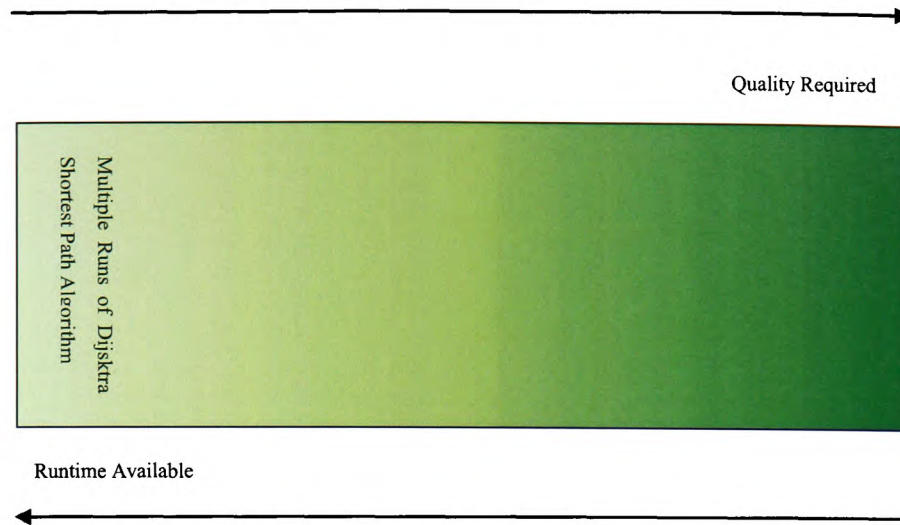


Figure 5.20 Highlighting Figure 5.16 with Multi Objective Dijkstra Approach

When considering the heuristic techniques considered for this work the optimality of their use over exact algorithms in the form of the SaA approach depends on the size of the graph being considered. On the smallest of the graphs, as shown in Figure 5.17 the SaA algorithm offers competitive levels of performance and completes a run in an average of 1.6 seconds compared with 1.8 for the Genetic Algorithm with K geodesics, 1.7 for the Simulated Annealing approach and 1.4 seconds for the PAES approach. The PAES algorithm outperforms the Skriver and Andersen (SaA) technique by 0.2 of second. However, the technique fails to return a complete set of optimal solutions even on graphs of this limited size (100 X 200). Given the limited success of the PAES algorithm the application of the technique is questionable. Figure 5.16 is updated to take into account a potential view of the techniques on small graphs in Figure 5.21. The absence of the Genetic Algorithm, the Simulated Annealing and the Tabu Search techniques from Figure 5.21 is deliberate given that the SaA technique outperforms those techniques in terms of run time and returns the exact set of optimal solutions. It is questionable in practice if on small graph sizes use of the PAES approach would be considered given the small difference in run time to the SaA and limited quality of the results obtained.

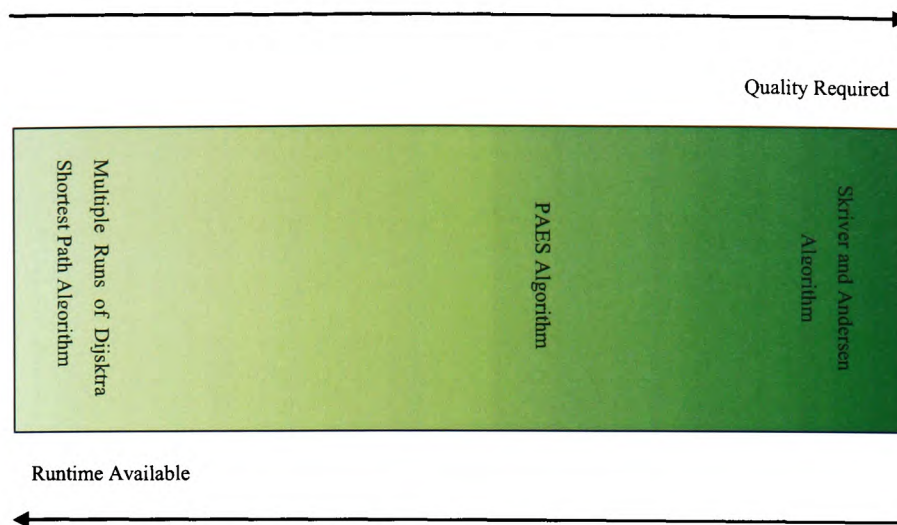


Figure 5.21 Run Approach on Small Graphs

When considering medium sized graphs the complexity of the MSPP is brought into sharp relief. Where on the smallest of the graph sizes an exact set of optimal solutions would be gathered using the Skriver and Andersen approach in a similar timeframe to an approximate set using the heuristics the same is not true when considering larger graph sizes, even those which may still be considered medium sized in scale. Of the heuristic methods the Tabu Search takes the longest to complete a processing run and yet still completes in approximately 18% of the time of the SaA approach. The Genetic Algorithm with random walking operates much more quickly than the Tabu Search technique and completes in 11% of the time of the SaA approach. The Genetic Algorithm with K Geodesic completes a run in 2.3% of the SaA approach. On smaller sized graphs use of the selected heuristic (the PAES approach) was questionable over the use of the SaA given the similarity of the runtimes seen. On medium sized graphs the appropriateness of the heuristic becomes clear. The Genetic Algorithm with K-Geodesics operates much more quickly than the SaA approach and importantly returns a high quality approximation of the optimal set returning a set of solutions of equal size to $PfTRUE$ or $|PfTRUE-1|$ in 98% of cases. There may be cases when the application of the SaA approach is appropriate and therefore that technique is included in Figure 5.22. The use of Dijkstra's shortest path algorithm is appropriate

where computational performance is considered more important than the optimal set of solutions.

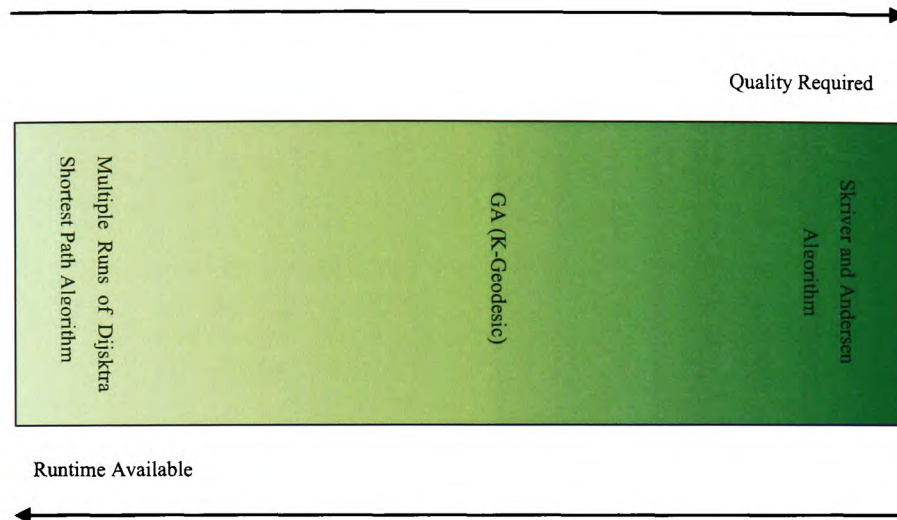


Figure 5.22 Run Model on Medium Sized Graphs

When considering larger graph sizes the performance advantage in terms of computational effort required of the Genetic Algorithm approaches over the other heuristic techniques becomes even more noticeable. The increase in runtime is highlighted in Figure 5.23. Figure 5.23 also reinforces the dramatic increase in runtime seen in the PAES and Simulated Annealing. Section 5.3 provides a rationale for the rapid increase in runtime seen in certain heuristics. Of importance from the graph is the relative stability of the Genetic Algorithm and the Tabu Search in comparison to the PAES and the Simulated Annealing approaches. In addition to both the relative stability in runtime the Genetic Algorithm is able to return more complete approximations of the optimal set on the larger graphs in comparison to the other techniques as demonstrated in Figure 5.24.

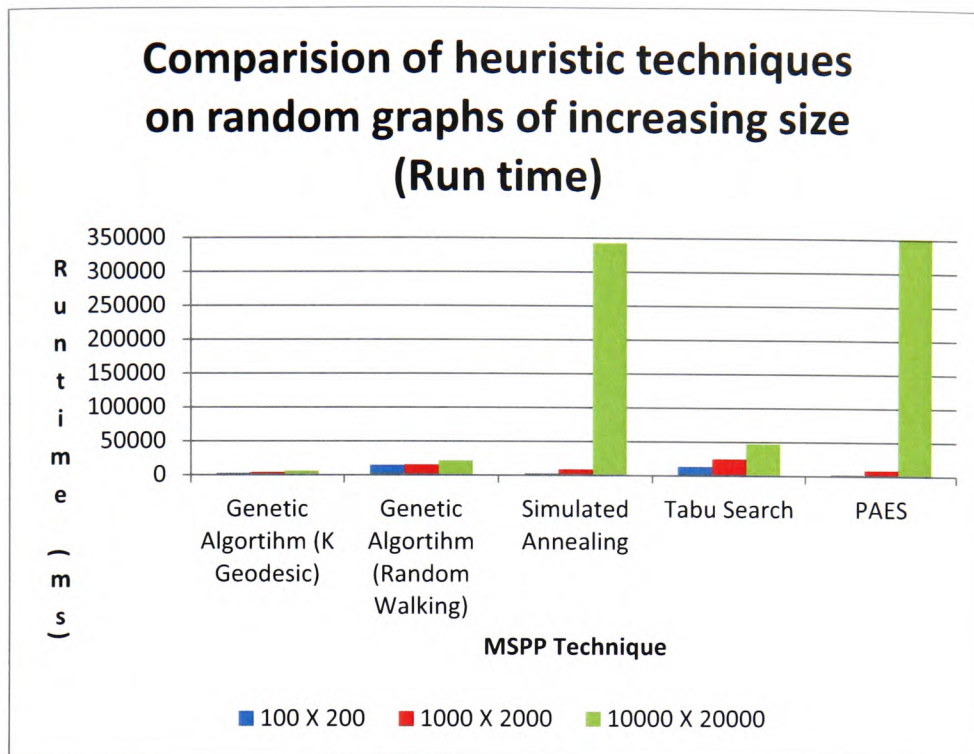


Figure 5.23 Comparison of Heuristics on Random Graphs (Run Time)

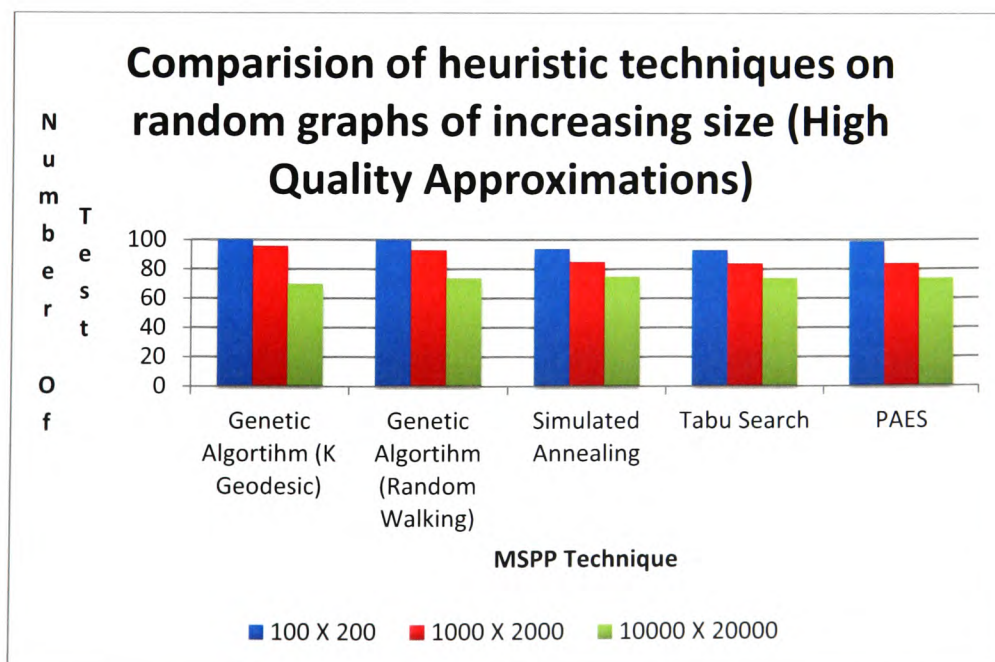


Figure 5.24 Comparison of Heuristic on Random Graphs (Quality)

Considering Figure 5.23 and Figure 5.24 jointly it seems logical to discount to application of the Simulated Annealing and PAES when considering the analysis of larger sized graphs. Those two techniques, in terms of quality, offer similar levels of performance as each of the other methodologies but exhibit much greater run times. The Tabu Search also offers a similar level of performance in terms of quality but a greater run time though the increase in the runtime seen is lower than that of the PAES and Simulated Annealing approaches. Therefore, for larger graphs the use of the Genetic Algorithm would appear the optimal approach to use. Figure 5.25 presents a usage diagram for the algorithms on larger graph sizes. Of the Genetic Algorithm approaches the K-Geodesic approach able to complete a processing run more quickly than the random walk and offer similar level of performance and therefore the choice of any of those two approaches may be considered appropriate. However, a question is present regarding the scalability of the K-Geodesic approach for graph sizes beyond those tested. The use of the K-Geodesic will logically require higher values in terms of population size and number of generations as the size of the graph under consideration grows which may affect the K-Geodesic approach more than the random walking approach to the genetic algorithm. In addition, comparing the increase in the runtime between the smallest (100 X 200) and largest (12000 X 24000) of the graphs used in the study highlights a much greater increase in runtime for the K-Geodesic (283%) than for random walking approach (66%). Where time is not a factor then logically the SaA method is still appropriate.

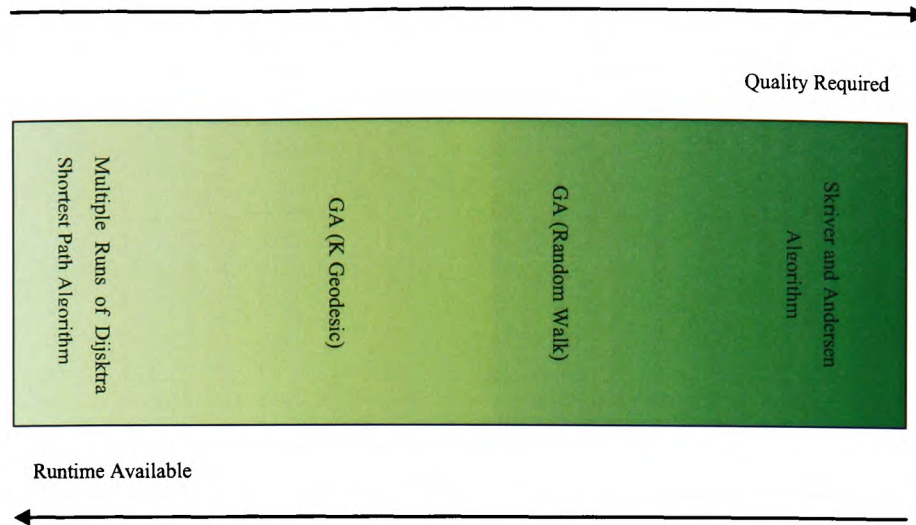


Figure 5.25 Run Model on Large Graphs

5.7.1. Summary Of Cross Algorithm Analysis

In summary, where a simplistic approximation of optimal solutions are required the approach of using multiple runs of Dijkstra shortest path algorithm is appropriate regardless of the size of the graph. The limitations of the technique are detailed at length elsewhere and will not be repeated here in the interests of brevity. On smaller sized graphs the exact algorithmic approach of Skriver and Andersen would appear to be appropriate. The algorithm offers a slightly higher runtime than that of quickest heuristic technique, the PAES approach, but has the advantage that it returns the exact and complete set of optimal solutions. For medium sized graphs then the Genetic Algorithm with K-Geodesic approach appears to be optimal in testing. For larger graphs the same appears to be true but a question remains regarding the graph sizes beyond those tested where the random walk approach appears to offer better levels of performance both in terms of scalability and, referring to section 5.5, quality. The section closes with that Algorithm 5.2 provides a possible selection model for the application of appropriate methods based on end user requirements.

Algorithm:	Simultaneous Performance Of Skriver and Andersen / Climaco and Martins MSPP approaches
Input:	S = The Identifier of the source vertex D = The Identifier of the target vertex $G = (V , E)$ = The Graph To Be Analysed $SLEEPERIOD$ = Amount Of Time to Wait Between Checks For Completion
Output:	$PfTRUE$ = Exact Set Of Optimal Solutions Between S And D
$P^{SA} = \{ \}$ // Set Of Optimal Paths Between S & D Using Skriver and Andersen Approach $P^{CM} = \{ \}$ // Set Of Optimal Paths Between S & D Using Climaco And Martins Approach T^{SA} = Start A New Processor Thread T^{CM} = Start A New Processor Thread Perform Skriver And Andersen Approach In T^{SA} Between S And D Perform Climaco And Martins Approach In T^{CM} Between S And D WHILE (T^{SA} Is Running And T^{CM} Is Running) { Sleep($SLEEPERIOD$) // Wait Until Either The Skriver and Andersen or Climaco And Martins Approach Have Finished } IF (T^{SA} Completed) $PfTRUE = P^{SA}$ IF (T^{CM} Completed) $PfTRUE = P^{CM}$	

Algorithm 5.1 Application of Both SaA and CaM Approaches

Algorithm:	Run Model Of MSPP Algorithms
Input:	<i>S</i> = The Identifier of the source vertex <i>D</i> = The Identifier of the target vertex <i>G</i> = (<i>V</i> , <i>E</i>) = The Graph To Be Analysed <i>GRAPHSIZE</i> = { <i>SMALL</i> , <i>MEDIUM</i> , <i>LARGE</i> , <i>VERYLARGE</i> } <i>PERFORMANCEREQUIRED</i> = { <i>FAST</i> , <i>APPROXIMATION</i> , <i>COMPLETE</i> }
Output:	<i>PfAPPROX</i> = Approximate Set Of Optimal Solutions Between <i>S</i> And <i>D</i>
<pre> IF {<i>PERFORMANCEREQUIRED</i> == <i>FAST</i>} { <i>PfAPPROX</i> = Extreme Points Of Optimal Front Gathered Using Dijkstra Approach Return <i>PfAPPROX</i> } ELSE IF {<i>PERFORMANCEREQUIRED</i> == <i>COMPLETE</i>} { <i>PfAPPROX</i> = Perform Skriver And Andersen Approach // Acquires <i>PfTRUE</i> Return <i>PfAPPROX</i> } ELSE // Approximation Algorithm { IF (<i>GRAPHSIZE</i> = <i>SMALL</i>) { <i>PfAPPROX</i> = Perform Skriver And Andersen Approach // Acquires <i>PfTRUE</i> Return <i>PfAPPROX</i> } ELSE IF (<i>GRAPHSIZE</i> == <i>MEDIUM</i> Or <i>GRAPHSIZE</i> == <i>Large</i>) { <i>PfAPPROX</i> = Perform Genetic Algorithm Using K-Geodesics Return <i>PfAPPROX</i> } ELSE IF (<i>GRAPHSIZE</i> == <i>VERYLARGE</i>) { <i>PfAPPROX</i> = Perform Genetic Algorithm Using Random Walk Return <i>PfAPPROX</i> } } </pre>	

Algorithm 5.2 Run Model of MSPP Approaches

5.8. Chapter Summary

The current chapter sought to review the algorithm in terms of the runtimes seen and provide a higher level analysis of the quality of the algorithms considering the optimal choice of approach. The methods employed can be seen in algorithmic approaches to the MSPP and the heuristic methods which have been developed. In addition the application of the Dijkstra shortest path algorithm was reviewed.

Hallam (2001) highlights that in many cases only the shortest path in each considered criteria may be needed for a decision maker to reach a viable solution as to the optimal path to choose. If that view is accepted as true then a methodology presented in this study (the calculation of shortest paths using the Dijkstra algorithm) shows that a series of solutions can be presented to the decision maker extremely quickly (around 555 ms on a graph sized 12000 x 24000). Where attempts are made to generate the complete Pareto optimal set a review of the runtimes for both remaining algorithmic methods demonstrate that the perceived difficulty in the MSPP problem is true. Both the Skriver and Anderson (2000) and Climaco and Martins (1982) approaches demonstrate an extreme increase in runtime as the size of the test graph is increased. It should be noted that in certain cases the Climaco and Martins approach is able to complete extremely efficiently. However, there is no way of predicating if such a scenario will be possible prior to any analysis being performed.

Each of the heuristic algorithms implemented is able to demonstrate a high degree of scalability with regard to the number of criteria under consideration. The calculation of the fitness of a solution, together with the extraction of the Pareto optimal fronts, is shown to be a relatively insignificant proportion of the runtime. The majority of the runtime can be seen to be occupied with the generation of candidate paths. Results indicate that on smaller graphs existing algorithmic techniques are best suited for the solution of the MSPP. As the size of the graph increases benefits seen in the use of heuristic approximations become more visible. A model based on the application of size of the graph together with a variety of user requirements has been produced.

Chapter 6: Behaviour Analysis

6. Behaviour Analysis

The previous chapter presented a comparison of the algorithms in terms of both runtime and quality. A model highlighting the possible usage of the various approaches where graphs of a certain size and performance requirements for the analysis was presented. The current chapter considers the behaviour of the various approaches when evolving towards the optimal set of solutions. The chapter starts by reviewing graph coverage achieved through the use of the random walk technique before considering the average lengths of paths seen in optimal sets compared with those produced by the random walk and the K-Geodesic approach. The chapter then considers the behaviour of heuristics individually, highlighting the effect of parameterisation on the algorithms.

6.1. Random Walking

The first phase of experimentation attempts to measure the effectiveness of random walking as a technique for exploring the graph. Emphasis is on the ability to generate a population of unique paths between a pair of randomly selected vertices on the graph. The outcome of the experiment is measured in the levels of graph coverage obtained by: a) a typical population set generated for a Genetic Algorithm and b) the levels of coverage obtained from a large number of random walks generated between two randomly selected vertices on the graph.

In an ideal world the task of finding optimal paths between two vertices would simply involve the generation of a set consisting of all valid paths between those vertices and then extracting the Pareto optimal set. A basic attempt at this approach has been made when attempting to identify the set P_{TRUE} . As has previously been discussed there is little realistic option when attempting to produce an accurate metric as to the actual number of paths between two vertices on a graph. The random walk is used

as a technique to overcome this, with the benefit that the random nature of the process should in theory allow a large degree of the search space, the graph, to be examined.

Figure 6.1 shows the a) number of unique edges and b) the number of unique vertices obtained by performing a series of 1000 random walks between 500 randomly selected vertex pairs. The chart provides a visualised view of what can be considered the ‘graph coverage’ and represent the amount of the search space (in this case a graph). Higher levels of graph coverage seen in higher levels of vertex and edge admission are required and indicate a higher proportion of the search space being considered. In the event that two directly connected vertices were selected that pair was removed from the analysis and another selected in its place. The graphs utilized in this study are generated randomly using the SPRAND software. Note that in the experiment the focus of interest is the ability of random walking to achieve graph coverage. As a result loops may be present in the solution paths.

In the following experiment an attempt is made to measure graph coverage that may typically be seen in a run of the Genetic Algorithm approach. The number of unique vertices seen during a typical run of the Genetic Algorithm is measured. The test is performed against the following parameter set values: a population of 30 and 30 generations; a population of 50 with 50 generations and finally a population of 90 with 90 generations. No repeating paths are allowed to enter the solution at any given generation. The figures represent the average graph coverage seen over 500 vertex pairings selected at random. The results are shown in Figure 6.2. All parameter sets selected will admit a large proportion of the graph into the search process. At the lower parameter set and on larger graphs around 40% of the graph vertex will be visited by the random walk process. Using the larger parameter sets the probability of a vertex being visited by the random walk is substantially higher; as is to be expected.

Figure 6.3 presents the number of unique paths that will be typically generated by the random walk process within the Genetic Algorithm approach with the parameter sets 30 X 30 (population 30, generations 30) on a variety of graphs. The graph highlights that the random walk technique is able to generate and admit a large number

of random paths between two vertices, a factor which will ultimately decide the success of the Genetic Algorithm and other techniques. The generation of a high number of walks indicates a high degree of coverage of the search space. Without the ability to adequately search the problem area then it is unlikely that the algorithms developed would be effective at solving the MSPP. A large percentage of the admitted paths discovered by the random walk algorithm are unique with the number of such paths increasing as the size of the graph increases. The experiment results are given in tabular form in Table 6.1.

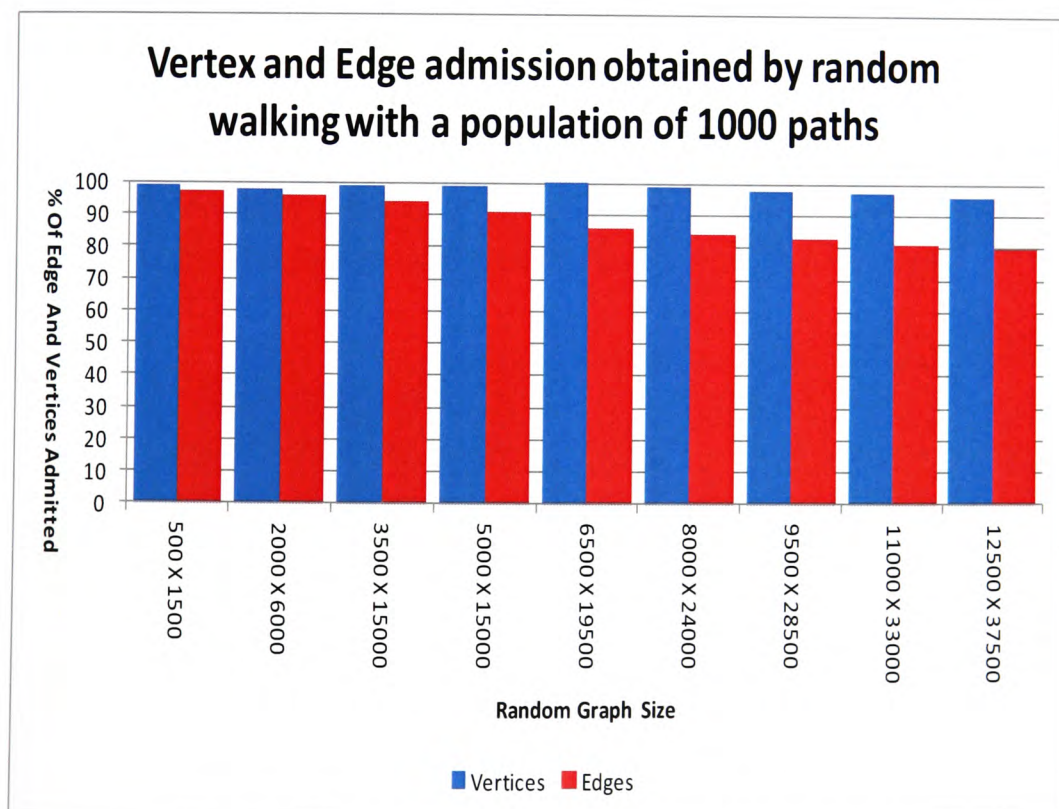


Figure 6.1 Graph Admission Rates Achieved by Random Walking

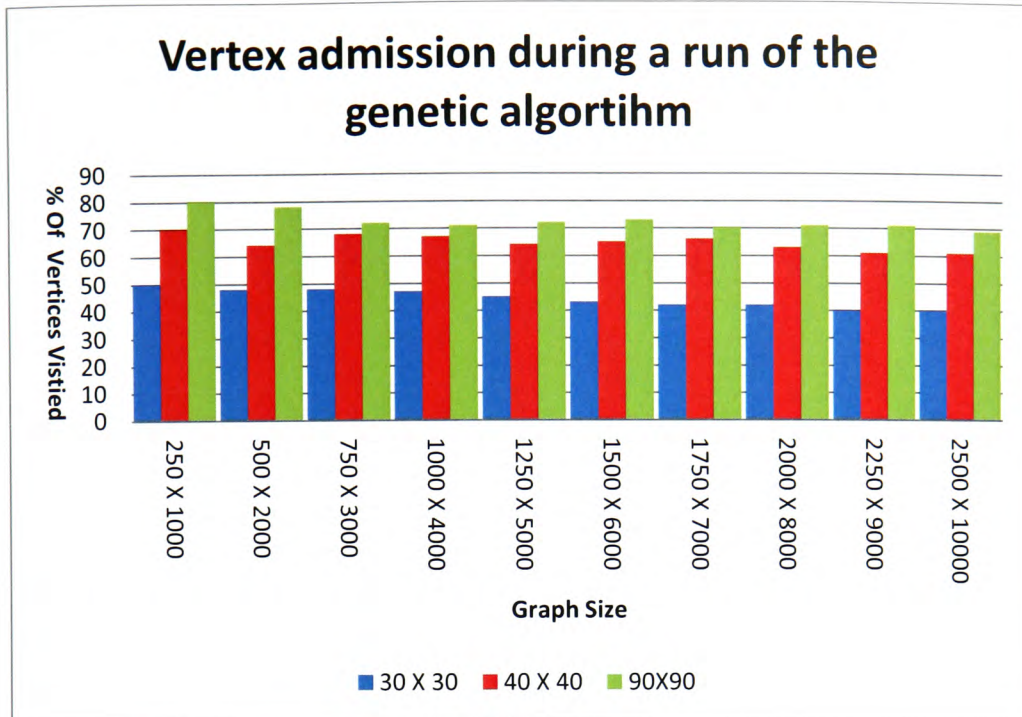


Figure 6.2 Vertex Admission During a Run of the Genetic Algorithm

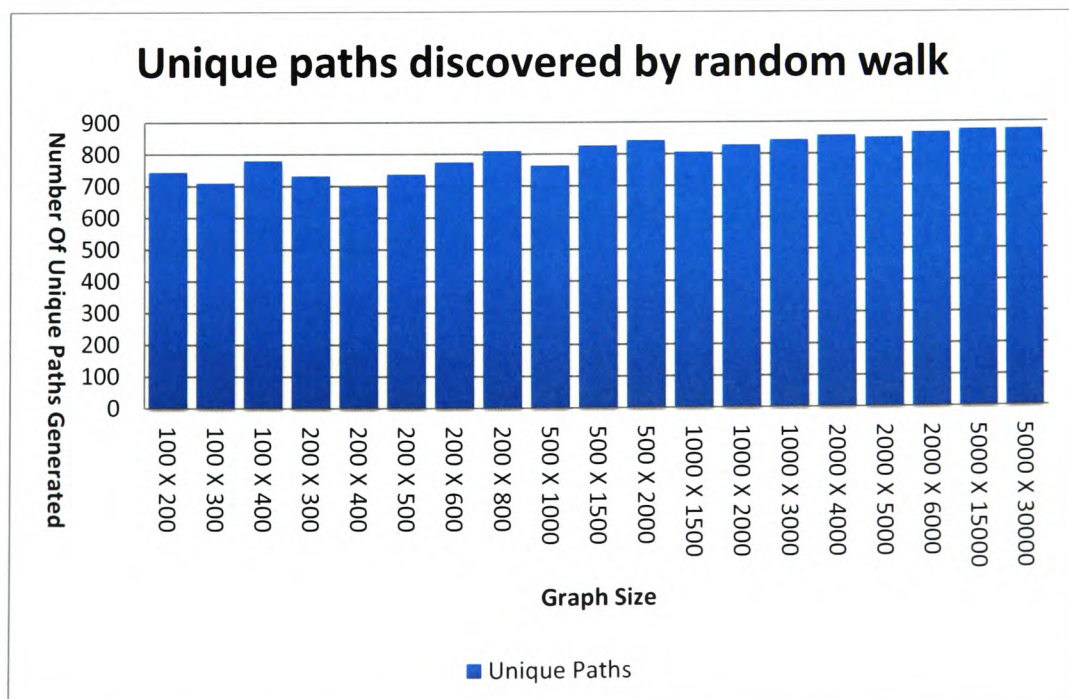


Figure 6.3 Unique Paths Discovered by Random Walk

Graph	Unique Paths
100 X 200	785
100 X 300	710
100 X 400	780
200 X 300	781
200 X 400	699
200 X 500	737
200 X 600	775
200 X 800	810
500 X 1000	764
500 X 1500	827
500 X 2000	843
1000 X 1500	806
1000 X 2000	828
1000 X 3000	844
2000 X 4000	858
2000 X 5000	851
2000 X 6000	867
5000 X 15000	876
5000 X 30000	877

Table 6.1 Number of Unique Paths Generated using Random Walk

The previous series of experiments sought to demonstrate the ability of random walking to achieve high levels of graph exploration. In the next the impact of the path generation technique on the typical path length is demonstrated. For the analysis 200 random selections of vertex pairings are made on a subset of the real world graphs. The sets *PfTRUE* for those vertex selections are generated using the technique outlined in the Chapter 4 with the averages edges being recorded. Figure 6.4 presents the average path length of each vertex pairing.

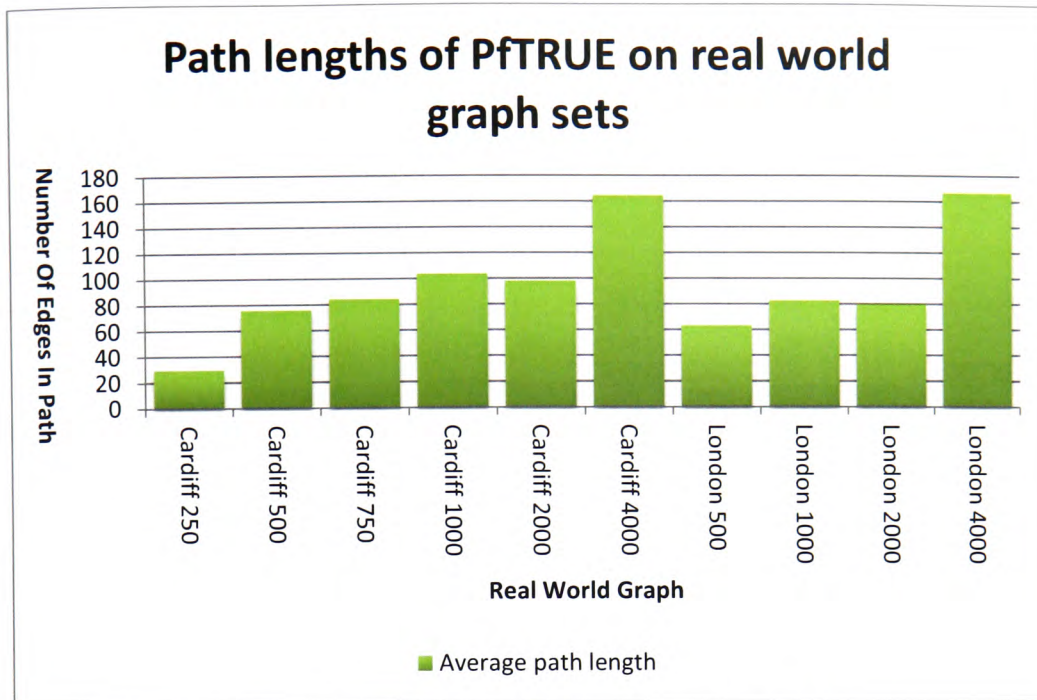


Figure 6.4 Length of Optimal Paths on Road Networks

Having acquired a general overview of the path lengths of the front *PfTRUE* a further analysis was undertaken into the average path lengths returned from both random walking and the K-geodesic approach. A typical run size of an evolutionary algorithm was selected; for the purposes of this experiment the parameter sets selected were 60 generations with a population size of 50. Such a parameter set would require the production of a *maximum* of 3000 unique paths. The same vertex pairings were selected as when reviewing the lengths of paths making up *PfTRUE*. Figure 6.5 gives the typical path lengths obtained using the K-geodesic approach. Figure 6.6 presents the results obtained using random walking.

The results appear at first glance to be something of a ‘mixed bag’, with each method exhibiting properties which in some ways show promise, yet demonstrate properties which may be seen as negative. Considering firstly the K-geodesic approach. It can be seen that the typical minimum path length returned is lower than that which may be seen in the set *PfTRUE*. This property is to entirely be expected given that the geodesic value is representative of the lowest number of steps between the two vertices. The negative aspects of the methodology arise when reviewing the maximum path

length values which for many of the data sets (such as Cardiff 4000 and London 4000) is lower than the maximum values of the paths making up $PfTRUE$. In comparison the result obtained using random walking demonstrate the opposite. The typical path length obtained using random walking is higher than that seen in the front $PfTRUE$ across minimum, maximum levels yet similar when considering the average length. It is hoped to demonstrate that the evolutionary algorithm approach will still function adequately for the task. Indeed, in Costelloe *et al* (2001) random walking is demonstrated to be able to provide an adequate base for the evolutionary approach. The same should apply to the K-geodesic approach, with the evolutionary operators making up for any data issues such as those highlighted above. The experimentation on random walking is concluded by highlighting that such data issues are to be entirely expected. If it were not for their presence then computing $PfAPPROX$ would be a simple matter of performing a series of random walks.

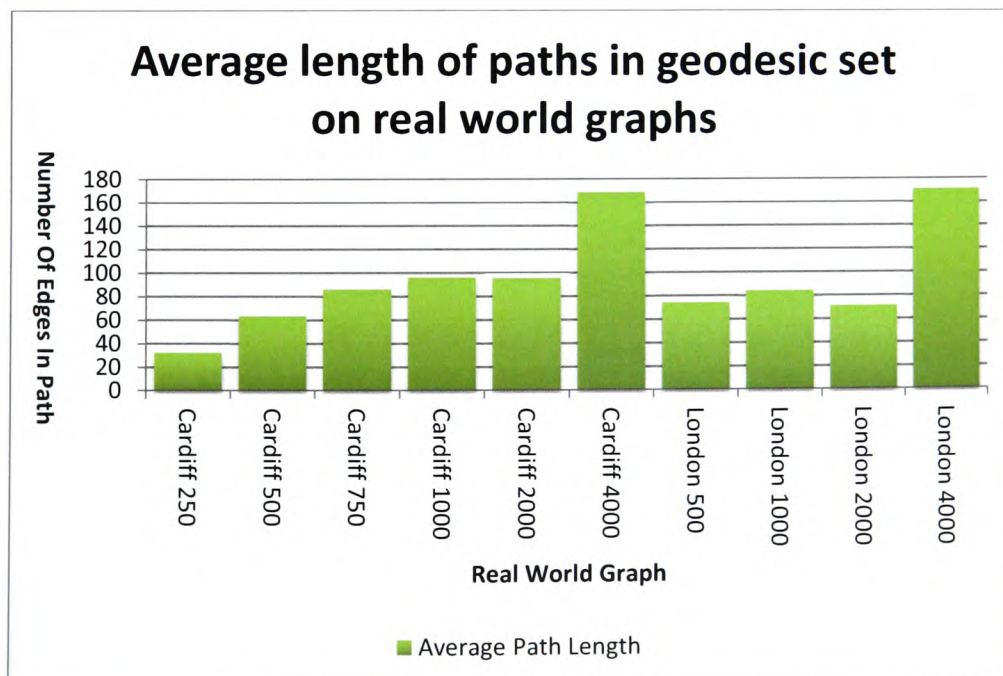


Figure 6.5 Path Lengths Obtained using K-Geodesics

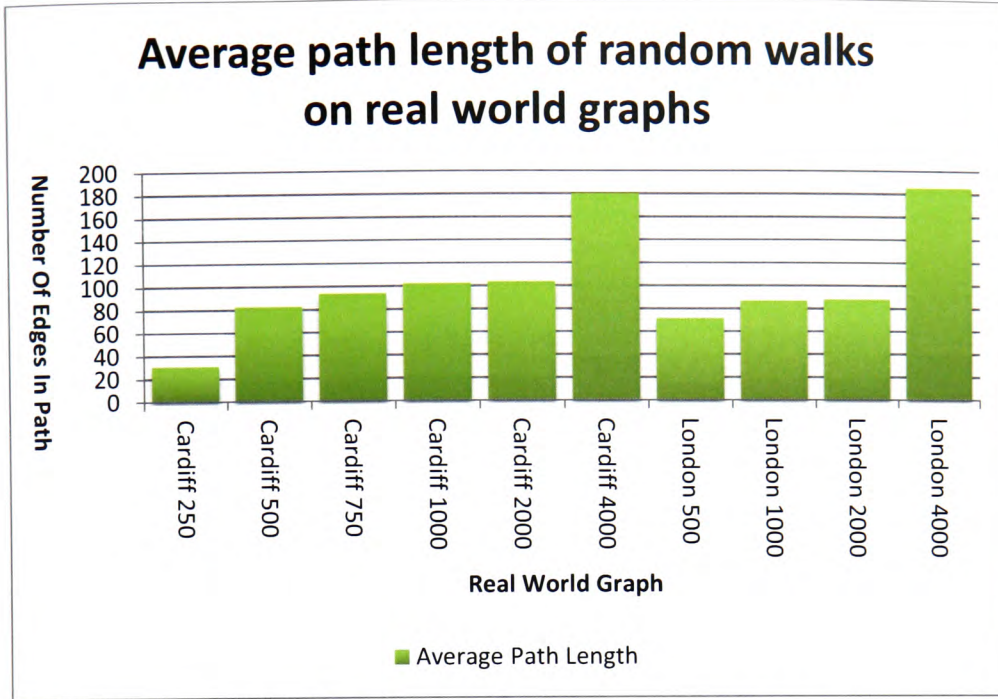


Figure 6.6 Path Lengths Obtained using Random Walking

6.2. Summary of Quality Tests on Heuristic Approaches

In this section an attempt is made to gather a quantitative overview of each of the heuristic algorithms abilities to approximate the front $PfTRUE$. For each heuristic, in this sense the genetic approach is considered as a heuristic technique, a series of one hundred tests are performed between two randomly selected vertices on a subset of random graphs. The same vertex pairings are used for each heuristic. The density of the graphs selected is approximated to that seen in real world graphs highlighted in section 4.2.1 with the edge/vertex ratio ranging from 1.5 – 4. In each of the result tables in this section three metrics are presented. Firstly where the quantity of solutions offered in $PfAPPROX$ is equal or greater than that of $PfTRUE$ ($|PfAPPROX| \geq |PfTRUE|$). Secondly where the difference is equal to $|PfTRUE| - 1$ (indicating that the approach failed to fully evolve with a single solution absent); and finally where the difference is greater than two or $|PfTRUE| - 2$.

6.2.1. Summary of Quality Tests on the Genetic Algorithm

For the genetic approach both the random walking technique and the K-geodesic approach are investigated. In the test the following parameters are set:

- population size of 80 for the random walk
- generation count of 80 for the random walk
- population count of 160 for the K geodesics
- generation count of 160 for the K geodesics
- the random walking technique uses a crossover rate of 0.35.
- the k-geodesic approach uses a crossover rate of 0.80.

The rates of crossover and mutation were initially taken from Mooney (2004). A brief period of experimentation for this study finds that increasing the levels of mutation beyond 0.1 has a negative effect on runtime with no discernible benefits in terms of result quality. Increasing the rate of mutation to higher levels may also have the effect of defeating the purpose of the mutation operation, that being to randomly introduce a change. If the rate of mutation is set too high the aim of the operation may be lost. In contrast the crossover operation is computationally cheap to implement therefore increasing the rate of crossover may be beneficial. The selection of parameters is one of the limitations of the experiments undertaken for the work as detailed in section 6.4, however it is believed that increasing the level of mutation would only have negative consequences while increasing the level of crossover would only have positive consequence. A period of empirical experimentation performed prior to the experiment indicates that a much higher rate of crossover is required for the K-geodesic approach to make a good approximation of the front Pf_{TRUE} hence the higher rate. Figure 6.7

presents the results gathered from the tests using the random walking technique while Figure 6.8 presents the results gathered using the K-geodesic approach.

The result of both experiments shows the Genetic Algorithm based technique to be very promising. A slight behaviour difference may be seen between the two approaches with the K-geodesic approach acting subtly differently at either end of the graph sizes. On smaller graphs (consisting of up to 500 vertices) the K-geodesic technique admits *slightly* more paths into *PfAPPROX* than the random walking approach although it should be highlighted that in itself the random walk is demonstrated over those same graphs to be a highly promising technique with the general admission rate of over 98% of Pareto optimal paths. Over the same graphs the K-geodesic admitted 100% although in practice the difference can be seen in only two of the graphs. In one of the two instances the random walk approach failed to identify a single path and in another graph where the random walk approach failed to find two paths in the approximate front. For that reason on smaller graphs the results obtained are comparable.

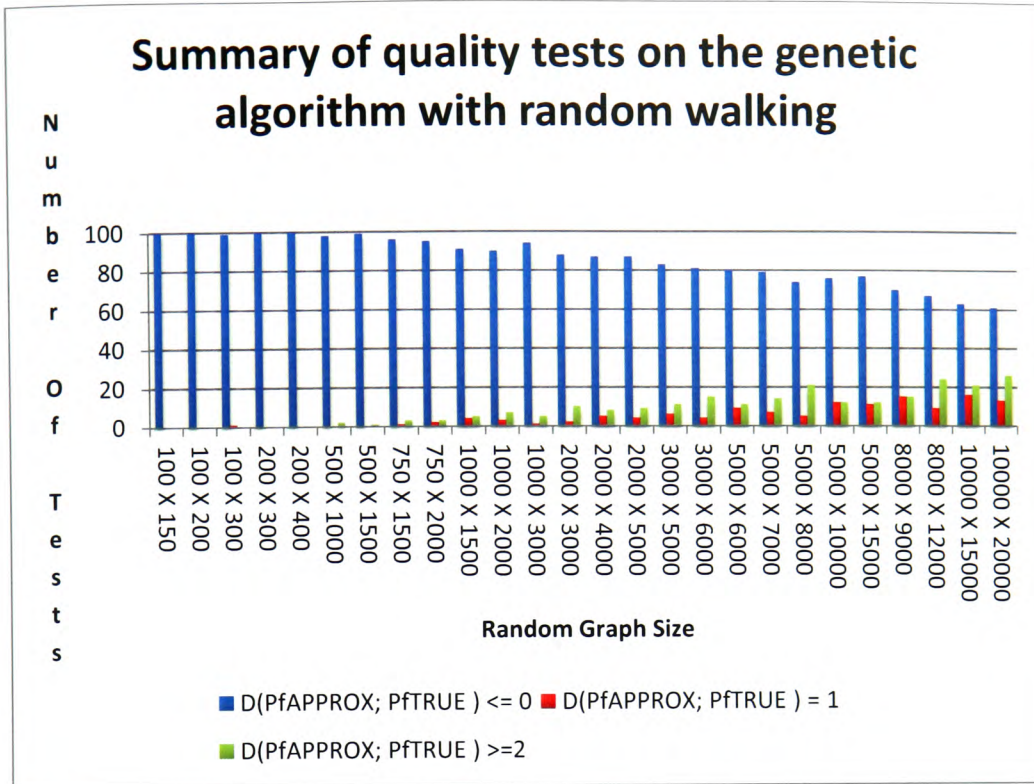


Figure 6.7 Summary of Quality Tests on Genetic Algorithm (Random Walking)

On larger graphs random walking is shown to be a more robust technique. The random walk based method still returns promising results with 74% of the tests either returning a set of the same size as $PfTRUE$ or only having a single solution less on the largest graph reviewed. The random walk was able to approximate the front $PfTRUE$ (in terms of size) in 61% of the tests compared with 52% as seen the K-geodesic approach. The K geodesic approach returns 70% of solutions have a front equal (or greater than) $|PfTRUE|$ or $|PfAPPROX-1|$.

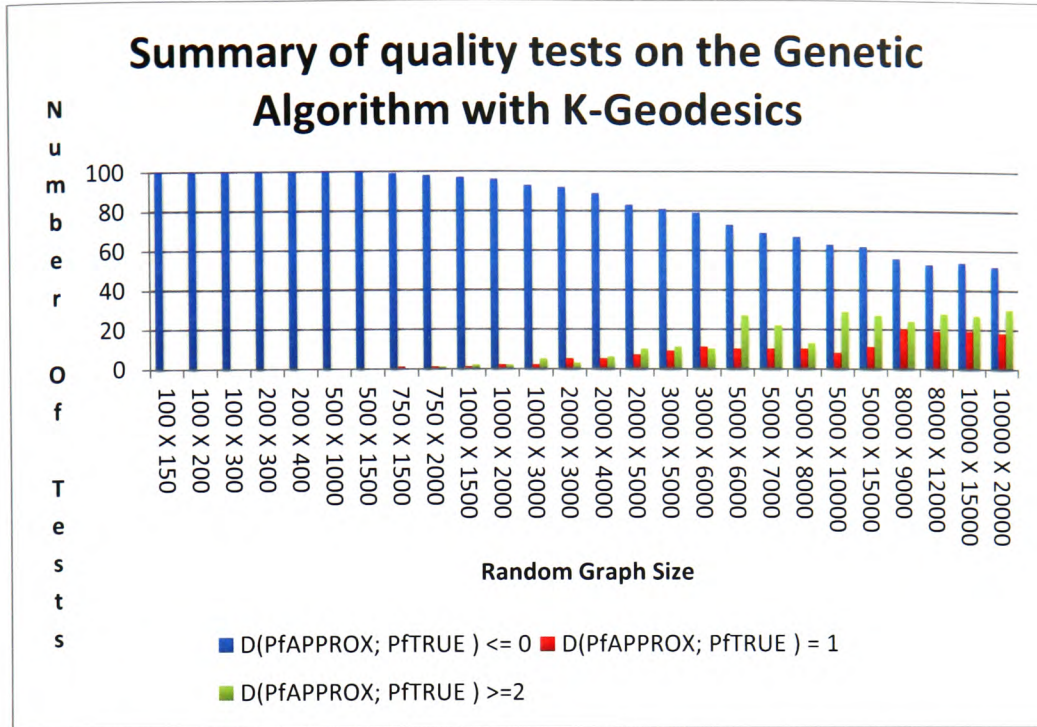


Figure 6.8 Summary of Quality Tests on Genetic Algorithm (K-Geodesic)

6.2.2. Summary of Quality Tests on the Tabu Search

In this section the results obtained from the Tabu Search are reviewed. For the purposes of the experiment the following parameters are used:

- 6500 iterations of the Tabu Search
- Tabu size of 150

The results (Figure 6.9) using the Tabu Search show less promise than the genetic approach on the larger sized graphs. In the case of the smaller graphs, those up to 500 vertices, the process is able to find either $PfTRUE$ or $PfTRUE-1$ in around 90-95% of cases as opposed to over 99% across the same graphs using the Genetic Algorithm. On the larger sized graphs the difference is more highly pronounced with a substantial size difference between $PfTRUE$ and $PfAPPROX$ being seen in 14 of the

selected cases. Typically the EA (RW) approach is able to identify a set $|PfTRUE| \leq |PfAPPROX|$ in 61% of cases. The Tabu Search is only able to do the same in 47% of cases on the largest of the graphs.

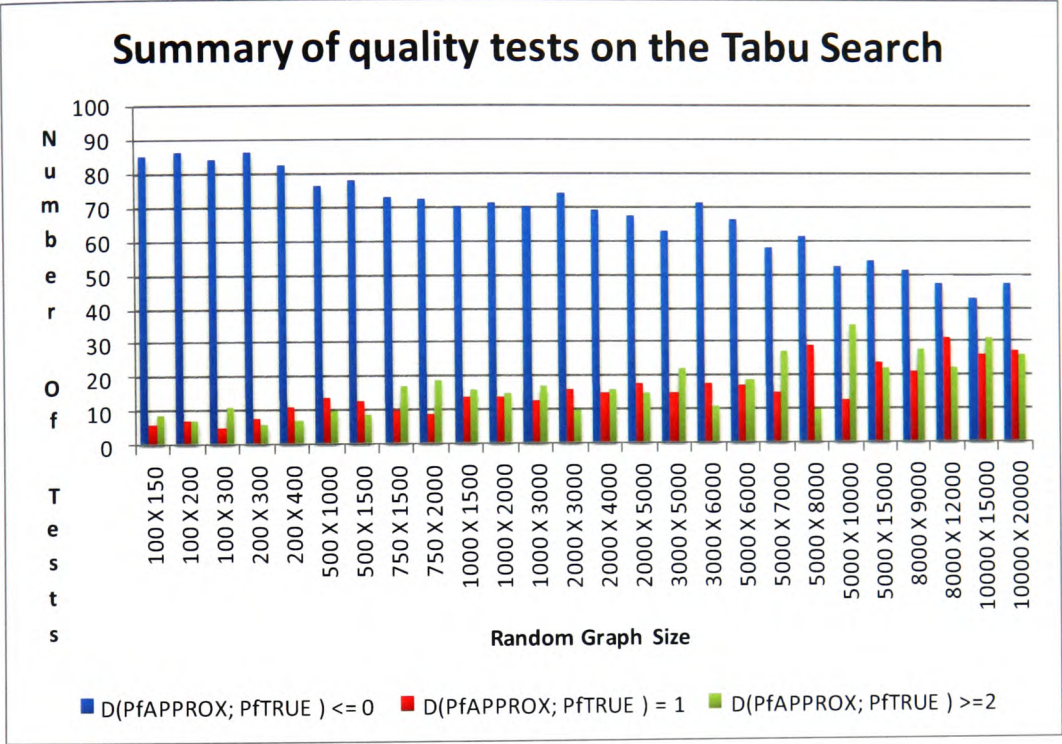


Figure 6.9 Summary of Quality Tests on the Tabu Search

6.2.3. Summary of Quality Tests on Simulated Annealing

For the Simulated Annealing tests the following parameter sets were used:

- a starting temperature of 125 is selected.
- the temperature is reduced by per anneal is 0.9997821
- the process is terminated when the temperature falls below 0.0001

The results from the Simulated Annealing algorithm are similar to those seen in the Tabu Search. In both experiments the methodology fails to identify a high quality approximation of the set $PfTRUE$ in a large proportion of cases. The results from the Simulated Annealing methodology show that for the smaller set of graphs then the results are slightly more promising than the results those obtained from the Tabu Search. However, when compared to those obtained from the Genetic Algorithm the results of the Simulated Annealing approach are considerably less promising, as were the results from the Tabu Search. Results from the smallest of the graphs demonstrate that the Simulated Annealing approximates $PfTRUE$ or $PfTRUE-1$ in around 96% compared with 91% in the case of the Tabu Search. The same can be seen in the largest of the graphs where the technique obtained a complete approximation of $PfTRUE$ in an additional two cases over the Tabu Search but remain less than the quantity obtained using the random walk which is able to identify up to an additional 12 complete cases of $PfTRUE$. Reviewing the results of the largest of the graphs indicates that Simulated Annealing and Tabu Search are more closely matched with the Tabu Search able to approximate $PfTRUE$ or $PFTRUE-1$ in only two tests less than Simulated Annealing.

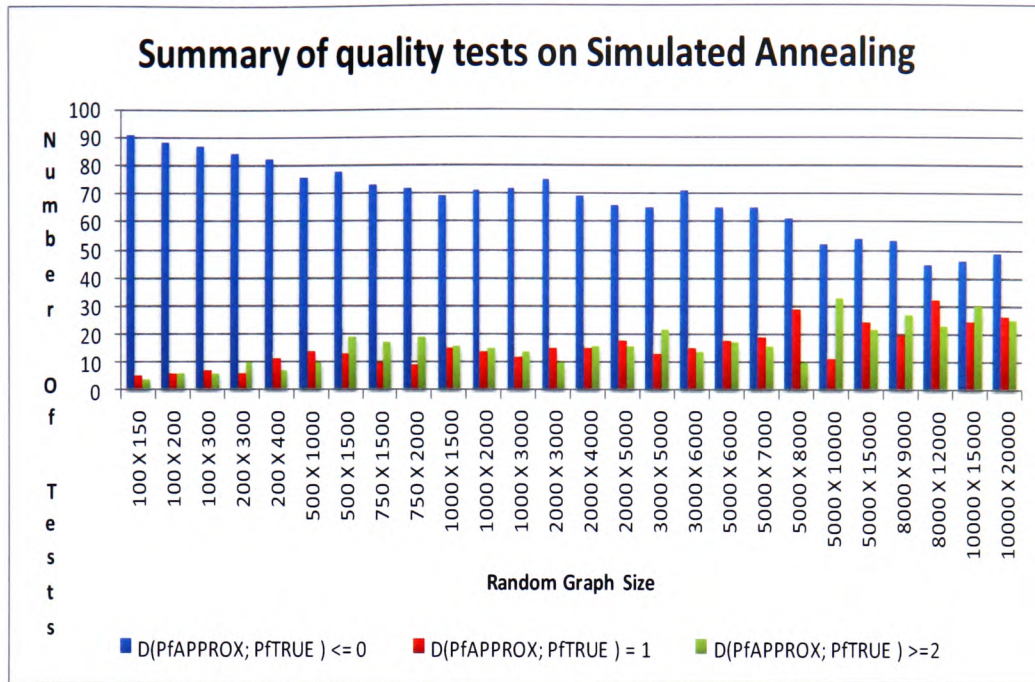


Figure 6.10 Summary of Quality Tests on Simulated Annealing

6.2.4. Summary of quality tests on the PAES

For the test on the PAES the number of iterations was set to 60,000. One interesting fact to arise from the tests of the PAES is the lack of effectiveness of the spread methodology for the MSPP. Due to the low number of solutions in the front $PfTRUE$ it can be seen that the technique has little practical value for the MSPP despite the fact that in many cases the front $PfTRUE$ may be clustered at either extreme with a concentration of solutions at either end of the front. The clustering of solutions is observed particularly on real world vertex pairings. In terms of the ability of the algorithms to identify $PfTRUE$ similar results to those other algorithms that can be considered 1+1 methodologies (a parent subjected to modification producing an offspring) are demonstrated. The summary of the results is provided in Figure 6.11.

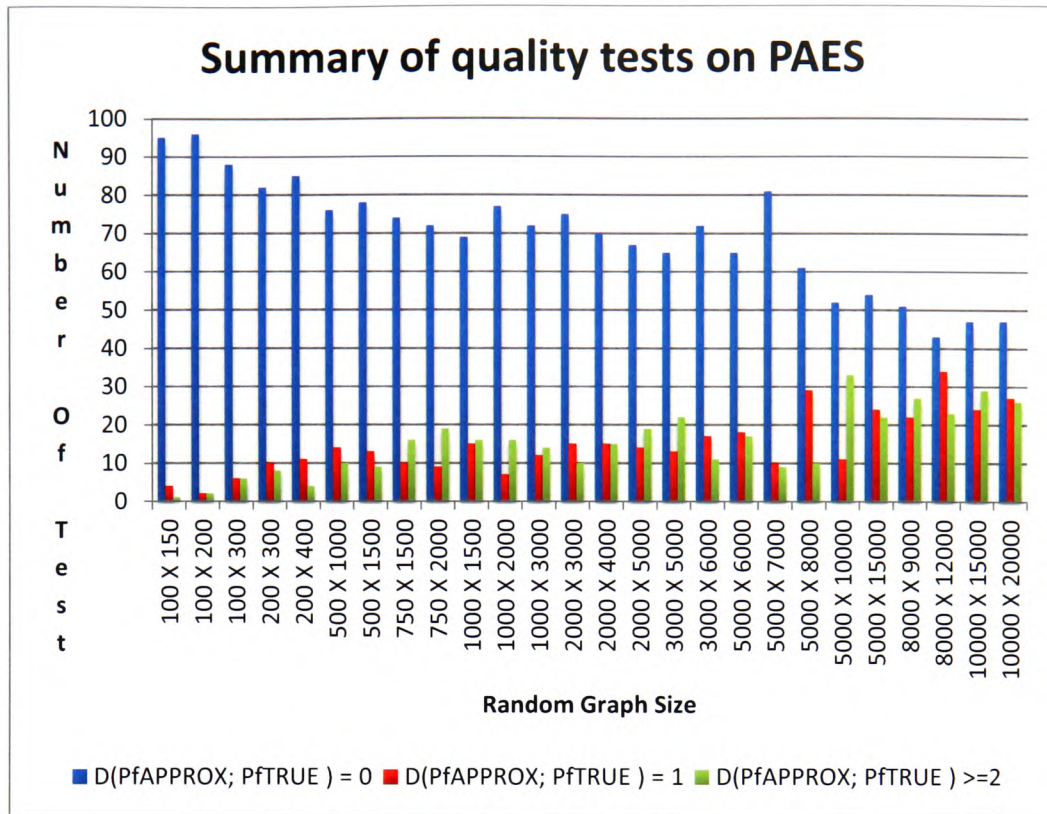


Figure 6.11 Summary of Quality Tests on PAES

6.2.5. Observations Gathered from the Quality Tests

A general observation that can be gathered from the experiment is that the 1+1 approaches are less successful than the population approach offered by the Genetic Algorithm at gathering a complete approximation of the set $PfTRUE$. A number of charts providing a visual presentation of the difference between the various heuristics has been provided in the previous chapter (Section 5.5). The choice of parameters should be reviewed; the aim of the experiment was to determine how well each of the algorithm types are able to produce a *good approximation* of the front $PfTRUE$ in a comparable time frame. The algorithms considered here are fully capable of, given enough time, producing a complete approximation of $PfTRUE$. The factors leading to this assertion have been discussed in Chapter 3. Whilst being aware of this fact the aim of the experiments was to carry out a more limited comparison. The experiments may have been allowed to continue until the $PfTRUE$ was fully calculated or an alternative

parameter set selected which may have presented the methodologies in a better light. An alternative would have been the automatic termination of the algorithms after a set period, such as five or ten minutes, and to analyse the quality of solutions gathered at that point. In those situations however the ability to carry out a comparable analysis would have been lost. The starting point for the parameter sets arise from similar work where genetic algorithms have been shown to perform well for the MSPP on similar sized graphs. A further factor that has to be considered when reviewing the result is that in many cases despite the fact that the ability of the algorithms to discover the complete set *PfTRUE* may be limited that does not necessarily equate to a poor performance indicator for the algorithms. It should be noted that in many cases where the set *PfTRUE* is fully acquired or only a single solution missing the results obtained from the 1+1 series of algorithms appear much more closely aligned with the Genetic Algorithm. Visualisation of Figure 5.12 in the previous chapter appear to confirm this. Considering graph 10000 x 20000 for instance. The PAES algorithm returns a value of 74% of cases where either the entire *PfTRUE* is acquired or only a single criteria is missing. This compares more favourable with the genetic algorithm based approach which demonstrate 70% acquisition levels using geodesics or 74% using random walking.

A final consideration is that in many cases the lack of convergence to the entire set *PfTRUE* may be considered a positive as opposed to a negative. In Chapter 2, various existing methodologies are highlighted in which the approach taken is to produce a subset of the *PfTRUE* front. The interactivity and utility function methodologies demonstrate this principle. In addition, Chapter 1 presented various route selection concepts where it was highlighted that route selection is often based upon personal preference. In that case, a subset of optimal solutions may be more than adequate.

6.3. Analysis of Algorithm Operation

The previous experiment dealt with how well the various algorithms converge towards the front *PfTRUE*. In this section, the aim is to consider a more detailed analysis into the behaviour of the algorithms with various behavioural patterns identified and subjected to a deeper forensic examination. As part of the experiment(s) the size and content of the front *PfTRUE* were captured at each iteration or generation of the algorithm. These were then subjected to review and analysis. The tests were repeated for a number of parameter sets in order to gauge the optimal parameterization required for the adequate solution of the MSPP.

6.3.1. Analysis of the Genetic Algorithm Operation

This section demonstrates how the Genetic Algorithm moves towards the optimal solution. In doing so four distinct scenarios are discovered and presented. Before detailing the exact scenarios the parameters used are reviewed. The tests are performed on the same pairs of vertices with a variety of genetic parameter sets. The sets are detailed in Table 6.2. In the case of the K geodesics approach the population and generation count are always double and use a crossover rate of 0.8 compared with 0.35 for the random walk.

Set	Population	Generations
A	30	30
B	30	50
C	40	50
E	50	50
F	60	50

Table 6.2 Parameter Sets Used in GA Analysis

It is interesting to note how the use of the K-geodesic path generation effects the general run time of the algorithm. Figure 6.12 presents the behaviour of the Genetic Algorithm when using the random walk path generation technique. In comparison the K-geodesic approach shown in Figure 6.13 shows the majority of processing is front-loaded in the processing run. After the generation of the initial population set, the algorithm performs quickly; the random walk technique however is spread equally through the run. However, Figure 6.12 and Figure 6.13 demonstrate that component of the Genetic Algorithm that requires the greatest computational effort is the generation of Candidate paths. The genetic operations, together with the set merge and evaluation schemes are in comparison computationally inexpensive and require an average of only 0.07 seconds per generation with a range of 0.02 seconds to 0.14 seconds. The behaviour is not limited to the Genetic Algorithm. In all the heuristics reviewed the path generation is where the majority of computational effort is undertaken.

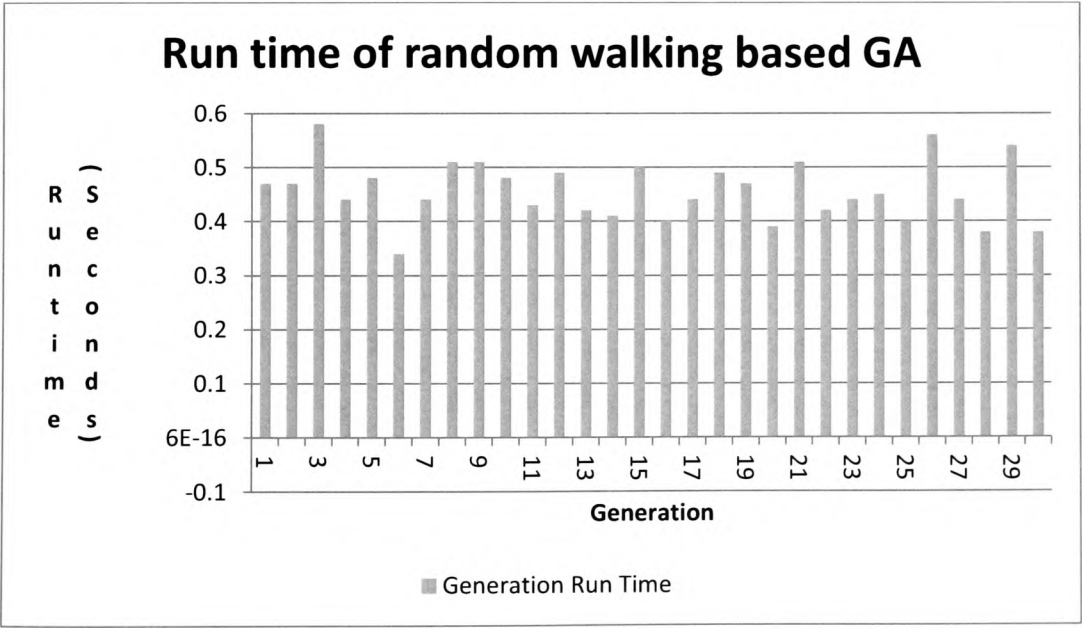


Figure 6.12 Runtime of Random Walking Based GA

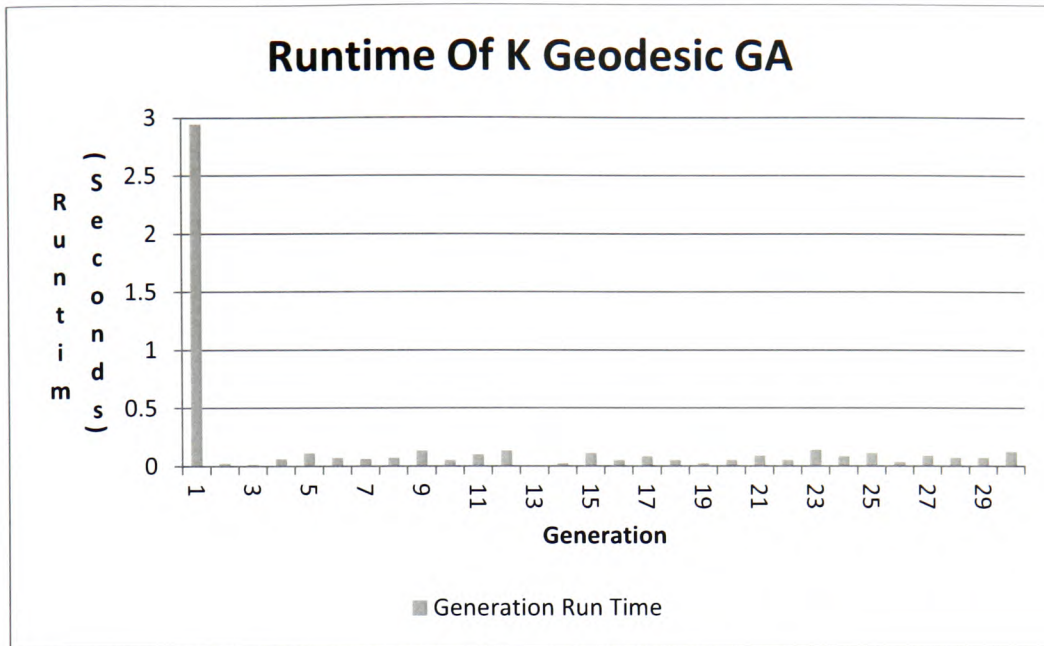


Figure 6.13 Runtime of K-Geodesic Based GA

Figure 6.14 and Figure 6.15 present the general outcomes of the experiment in terms of the ONVGR metric. This gives the ratio of solutions present in *PfTRUE* also present in *PfAPPROX*. Figure 6.14 presents the results of the test when the experiments are performed on 2D graphs (2 criteria) while Figure 6.15 presents the results obtained from 3D graphs. Sets A-E corresponds with the parameter selection given in Table 6.2.

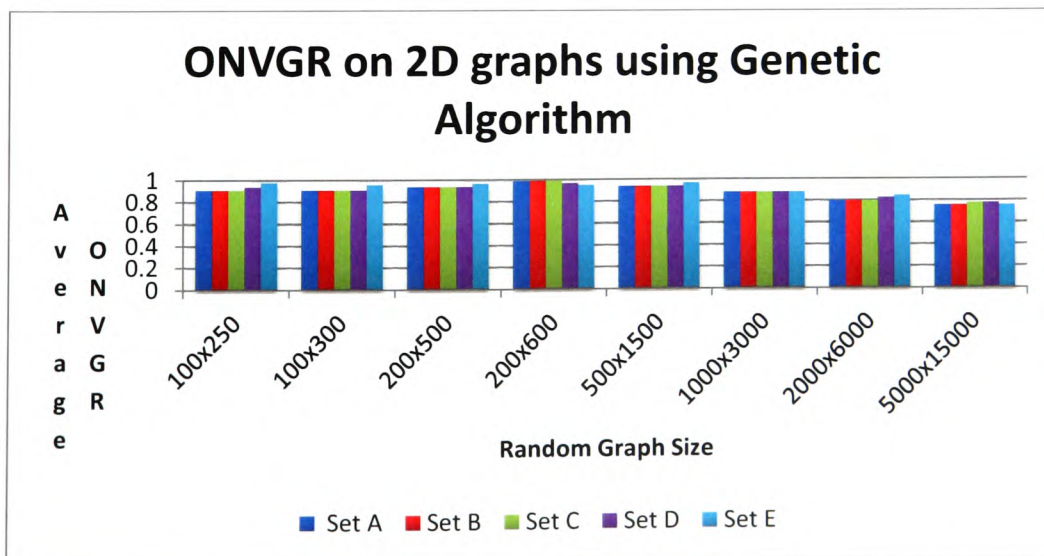


Figure 6.14 ONVGR on 2D Graphs Using the GA (Random Walking)

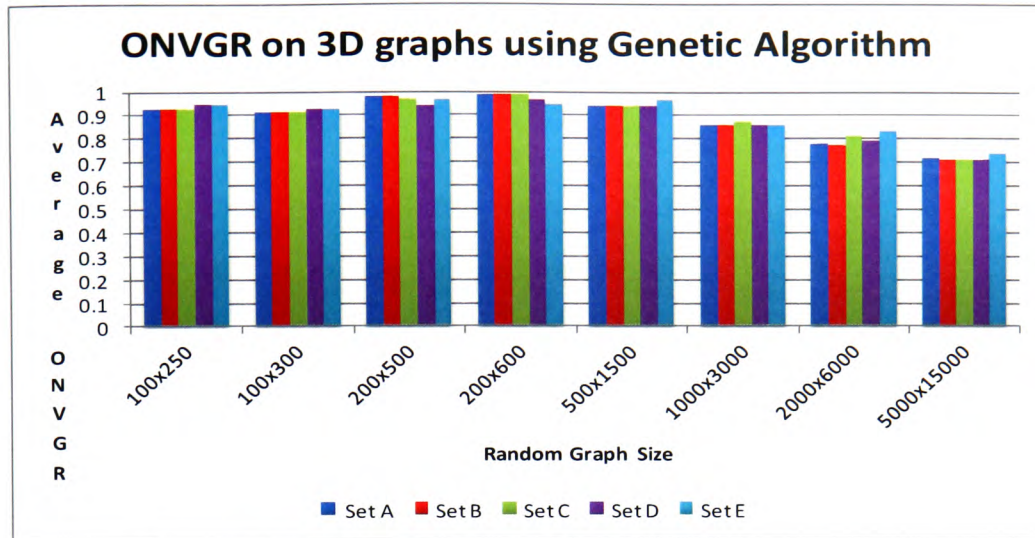


Figure 6.15 ONVGR on 3D Graphs Using the GA (Random Walking)

The tables contain the average ONVGR obtained from the Genetic Algorithm when applied to 100 randomly selected vertex pairs for which the front $PfTRUE$ has been calculated using a brute force technique based upon the K shortest path described in the previous chapter. Taking the results from Figure 6.14 and Figure 6.15 alone may give raise to the perception that the Genetic Algorithm is performing poorly; however, that is not necessarily the case. Considering the following case study identified:

Between the vertices $\{73, 3455\}$ on graph size 5000x15000, brute force analysis indicates that the front $PfTRUE$ will consist of 12 solutions. The Genetic Algorithm presents a $PfAPPROX$ consisting of four solutions obtained using Set A before raising to 7 solutions when the algorithm is performed using Set E. Giving a ONVGR ranging from 0.33 to 0.58. The algorithm has generated a high number of optimal paths, which may be confirmed through visual inspection of the two fronts. The ONVGR ratio is, however, poor.

When Figure 6.7 is reviewed in conjunction with the ONVGR (Figure 6.14) then the Genetic Algorithm is shown to be much more promising. The number of solutions typically offered in the front $PfTRUE$ tends to be limited with upper bounds on the random graphs of 14. Therefore, the Genetic Algorithm failing to discover even a small

number of solutions, say two, three or four, has a dramatic effect on the ONVGR. The fact that the values demonstrated in Figure 6.14 and Figure 6.15 are as high as they are shows a great deal of promise.

6.3.1.1 Observations of the Genetic Algorithm Operation

This section attempts to demonstrate how the genetic algorithm moves towards the optimal solution. In doing so four distinct scenarios are discovered and presented. In the first of these four scenarios the Genetic Algorithm quickly evolves to a good approximation of the set $PfTRUE$. The second of the four scenarios highlights an increase in the quality of the front $PfAPPROX$ during the course of an operational run. In the third scenario initial generations yield a poor approximation of the front $PfTRUE$ before becoming more complete. In the fourth of the scenarios an attempt is made to establish that the Genetic Algorithm may not always return what can be described as a ‘good’ scenario although as previously described the nature of the problem in the real world would prove to make such an assessment a difficult proposition. The quality of the solutions has been measured using a number of quality metrics (ONVGR, Generational Distance and Error Ratio) and visual comparison of the front $PfTRUE$ and $PfAPPROX$.

6.3.2. Analysis of the Tabu Search Approach

In this section the results obtained from the Tabu Search are reviewed. The same vertex pairings are selected, having been stored as part of the previous experiments. The size and contents of the front *PfTRUE* were recorded upon the completion of each iteration of the Tabu Search. For the purposes of the experimental phase the following numbers of iterations were selected:

- 1500 Iterations (Set A)
- 2500 Iterations (Set B)
- 3000 Iterations (Set C)
- 5000 Iterations (Set D)
- 7500 Iterations (Set E)

The experiment was repeated a number of times with a variety of Tabu list sizes ranging from 25 through to 500 with increments of 25, i.e. 25, 50, 75 etc. In the interests of brevity, not all results are included, though general observations are raised where needed. In the results highlighted, the Tabu list size is set to 400. Figure 6.16 presents the ONVGR metric values on a series of random graphs consisting of two criteria. Figure 6.17 highlights the effect on the ONVGR of increasing the size of the Tabu list on the graph 5000 X 15000 using the parameters in Set D (5000 iterations).

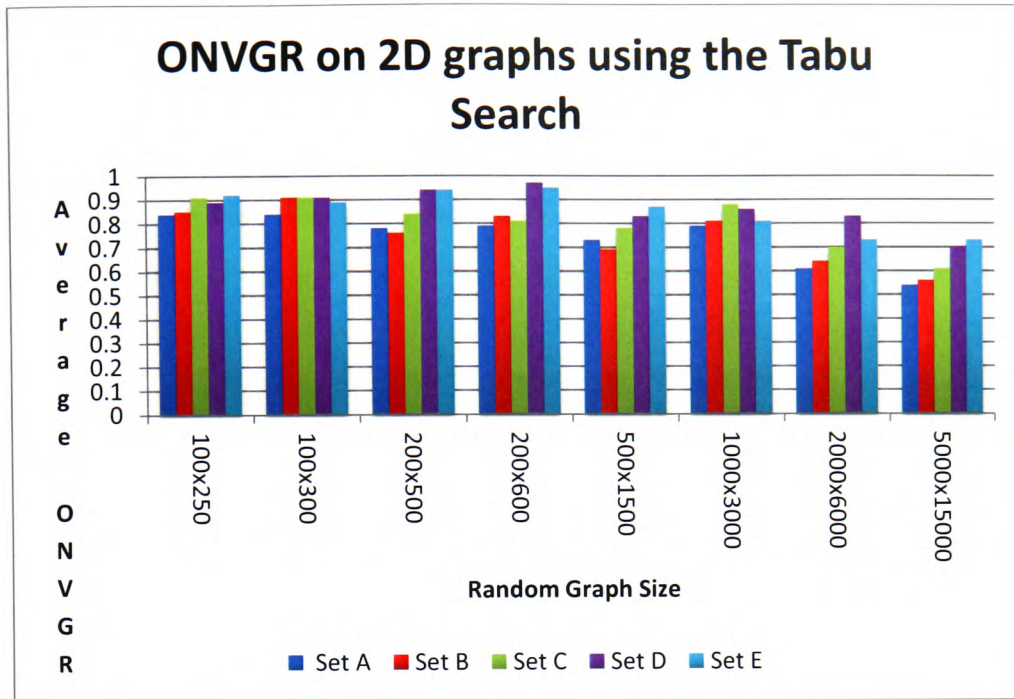


Figure 6.16 ONVGR on 2D Graphs Using the Tabu Search

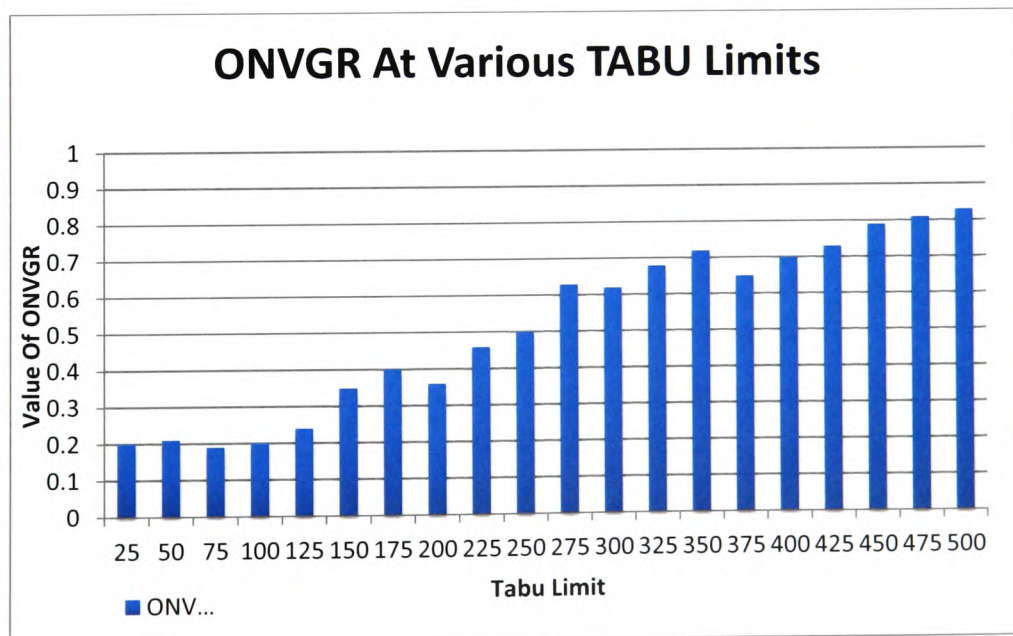


Figure 6.17 Effect of Increasing Tabu List Size on ONVGR (Set D)

6.3.2.1 Observations of the Tabu Search Operation

Upon the completion of the experiments on the Tabu Search algorithm the outputs of each vertex pairing were reviewed. The performance of the algorithm depends heavily on the size of the Tabu list. In those cases where the list is set to a low value (0-100) the algorithm regularly fails to capture a high quality approximation of the front $PfTRUE$. In many of the tests performed in such cases the algorithm often includes results in the set $PfAPPROX$ that are locally but not globally optimum. Increasing the size of the Tabu list decreases the frequency of locally optimal solutions being seen but does not entirely remove their presence. Figure 6.18 demonstrates how the size of front $PfAPPROX$ varies as the algorithm moves through the iterations in an example run. Figure 6.19 presents the front $PfAPPROX$ returned from the algorithm when compared with the front $PfTRUE$ in the same run as highlighted in Figure 6.18. In the example given in Figure 6.18 and Figure 6.19 the number of iterations performed is 1500 (Set A) when the Tabu size is set to 25 and the graph is 1000 x 3000 with two criteria being considered.

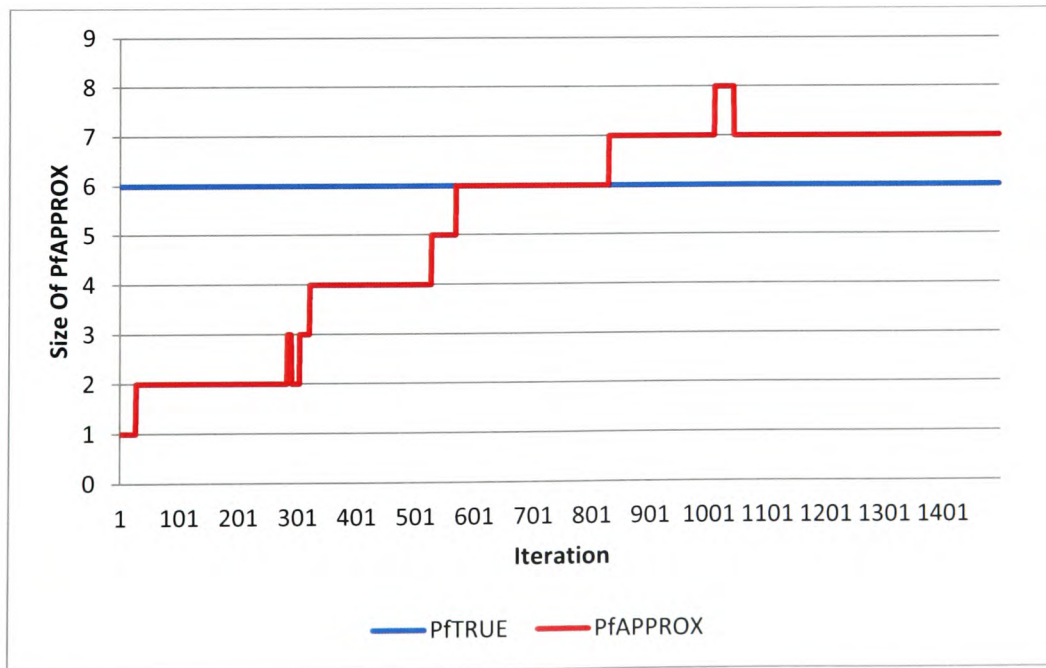


Figure 6.18 Example of $PfAPPROX$ Containing Non-Optimal Solutions

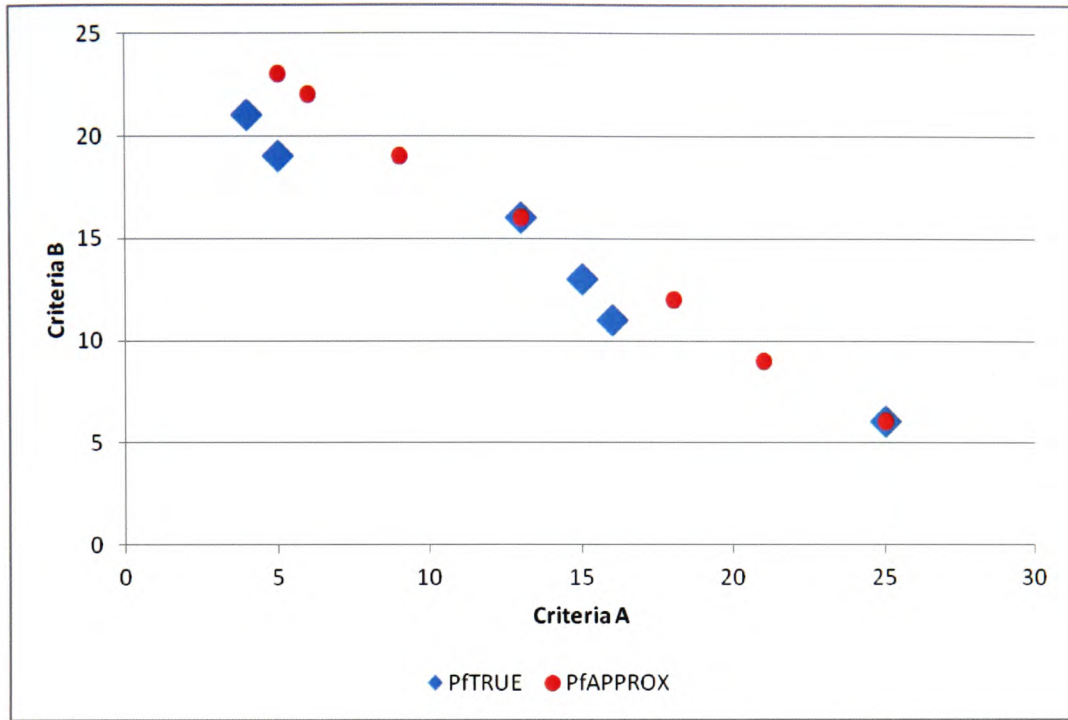


Figure 6.19 Example Fronts $PfTRUE$ & $PfAPPROX$ Obtained using Tabu Search

In the example just two of the seven solutions making up the front $PfAPPROX$ are also present in the front $PfTRUE$. The remaining five solutions are not present and represent locally optimal solutions. As the size of the Tabu list increases the ability of the algorithm to both capture an accurate approximation of the $PfTRUE$ increases while the presence of locally optimum solutions decreases. The analysis of the tests using the largest Tabu value (500) highlight that non-globally optimal solutions are less commonly present being seen in around 28% of cases when compared to 43% of cases where the Tabu list size is set to 100. Figure 6.20 presents an example of such a case taken from graph 500x1500 with two criteria being considered. In terms of the ONVGR the result appears to be poor given that the front $PfTRUE$ consists of five solutions with the set $PfAPPROX$ consisting of three solutions, one of which is not globally optimal, although as seen in Figure 6.20 the locally optimal solution is close to the front $PfTRUE$.

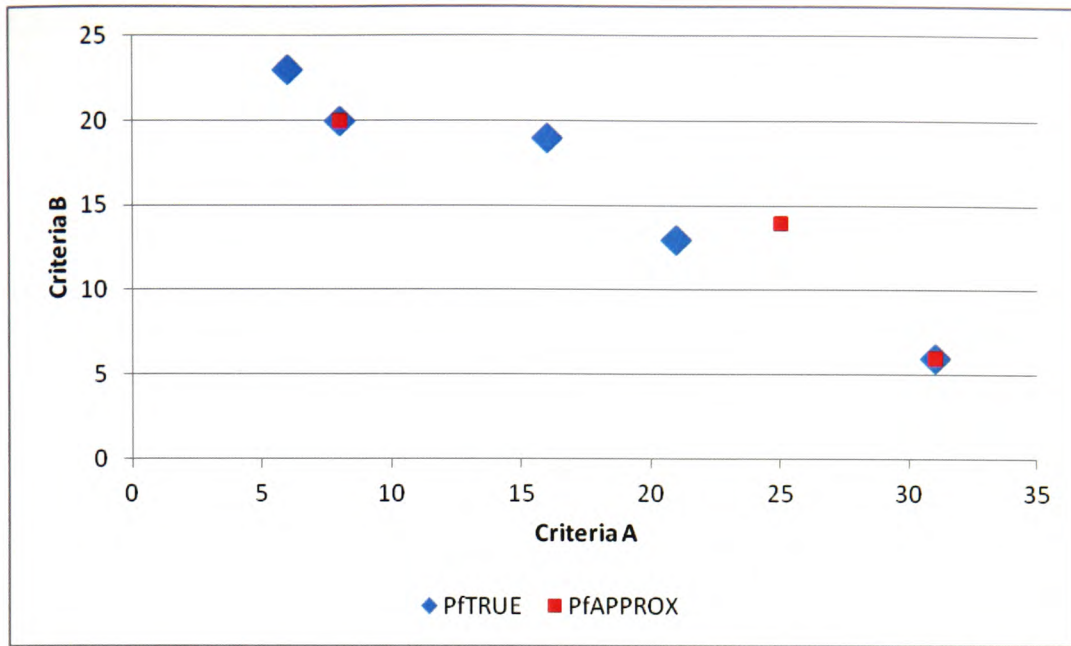


Figure 6.20 Comparison of *PfTRUE* & *PfAPPROX* With Locally Optimal Solutions

Locally optimal paths are regularly found at the start of the Tabu Search run. As the algorithm begins to discover globally optimal solutions then the locally optimal solutions go through a gradual pruning process. Where the algorithm is able to identify a higher number of members of *PfTRUE* then there is less likelihood of locally optimal solutions being present. Where locally optimal solutions are present then the solutions provided may still be valid. Locally optimal solutions can be seen in 46 of the solutions on graph size 1000 x 3000 using a Tabu size of 100. For 26 of these 46 solutions the generational distance metric reports an error value of less than four indicating that although local optimal solutions are present the solution is still a reasonable approximation of *PfTRUE*. The solutions provided in Figure 6.19 and Figure 6.20 may be of value to a decision maker.

6.3.3. Analysis of the Simulated Annealing Algorithm

In this experiment, the performance of the Simulated Annealing algorithm is reviewed. Table 6.3 gives the starting temperature and decline rates used in the experiment. Figure 6.21 gives the ONVGR metrics presented by the algorithm. It may also be useful to view the results given in Figure 6.21 in conjunction with those given in Figure 6.10. The purpose of Figure 6.21 is to provide a basic review of the ability of the algorithm to capture an approximation of $PfTRUE$ measured using the ONVGR metric. In 14 cases as shown in Figure 6.10 the Simulated Annealing approach failed to find over two solutions on graph sized 1000 x 3000. The observation review of simulated annealing provides examples of what may be an extremely poor solution. It should be noted that such results are often seen in lower parameter sets and on larger graph for the other heuristics in addition to the Simulated Annealing algorithm. Such scenarios are however more prominent for the Simulated Annealing and PAES approach. The example is one that can be considered a worst case scenario.

Parameter Set	Starting Temperature	Decline Rate
Set A	100	0.9999301
Set B	120	0.9995521
Set C	120	0.9996421
Set D	120	0.9997101
Set E	120	0.9997721

Table 6.3 Simulated Annealing Parameters

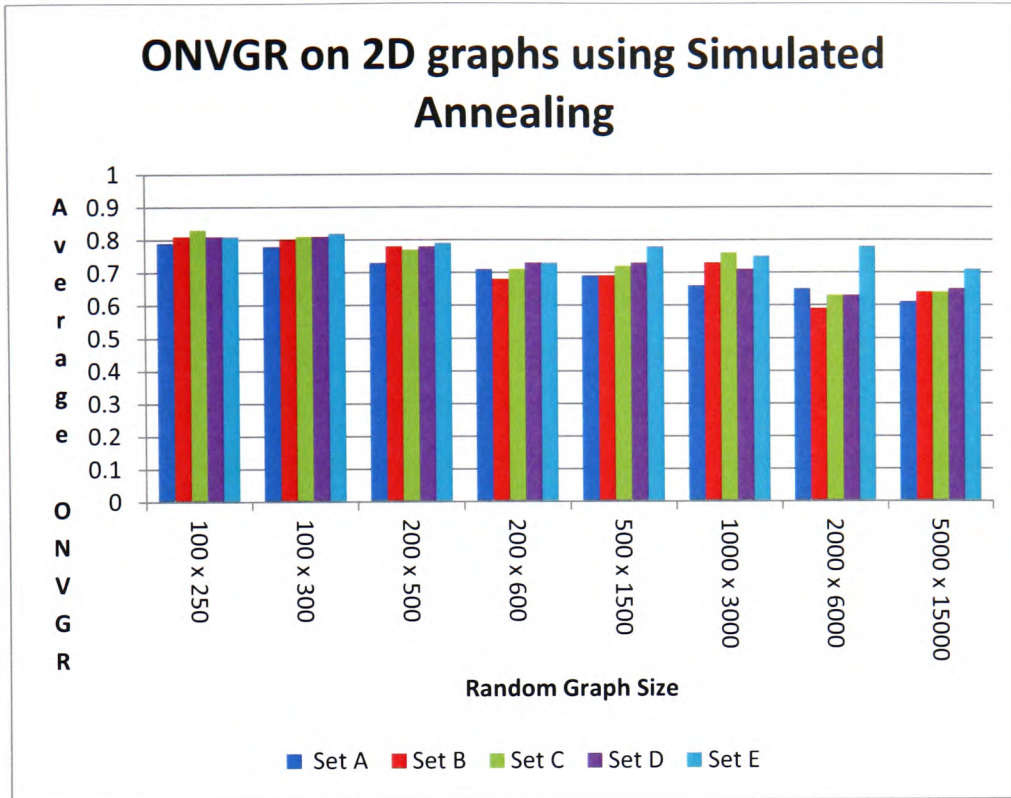


Figure 6.21 ONVGR on 2D graphs Using Simulated Annealing

6.3.3.1 Observations of the Simulated Annealing Operation

In a previous section (6.3.2.1) a practical example, shown in Figure 6.19, of a scenario is given where the Tabu search methodology returns what can be considered a very poor approximation of the front Pf_{TRUE} was given. Here a similar situation regarding the Simulated Annealing algorithm is considered. The graph size is 5000 x 15000 with two criteria under consideration. There are 11 solutions in the set Pf_{TRUE} with only two returned in the set Pf_{APPROX} . Figure 6.22 presents a visual comparison between the two sets.

Figure 6.23 presents the general performance of the algorithm as it proceeds through the run. The basic behaviour is that the first path will be judged optimal and so it is added to the external archive. The algorithm then quickly determines a second locally optimal solution before entering a quiet period where little changes. The algorithm then finds a further two locally optimal solutions before finding a member of *PfTRUE* reducing *PfAPPROX* to a single member solution. A short while after this the algorithm finds a promising yet locally optimal solution. Few variations can then be seen until a second locally optimal solution is found. This is quickly followed by the identification of a second member of *PfTRUE* removing a locally optimal solution. No further changes are seen for the remainder of the algorithm run. The result is an ONVGR of 0.18. There are six instances seen using simulated annealing where what may be considered very poor results can be seen as in the example, i.e. the simulated annealing algorithm returning a very low approximation of *PfTRUE* consisting of at most two solutions resulting in an ONVGR of less than 0.2 for those six cases. The examples can largely be seen on larger graph sizes. In such cases locally optimal but high quality solutions, as given in Figure 6.19, may be considered a very good overall solution despite the presence of those locally optimal solutions.

Table 6.4 highlights a trend that can be seen throughout the performance of the Simulated Annealing algorithm. It gives the number of tests where the count of solutions making up *PfTRUE* falls within a specific range. It then gives the number of cases where the Simulated Annealing algorithm fails to find two or more solutions. The final column gives average number of solutions for each of those cases found using parameter Set C. The figure in brackets gives the maximum value that could be achieved based on the contents of *PfTRUE*. For the parameter set chosen there are thirty solutions where the difference between *PfTRUE* and *PfAPPROX* is equal to or greater than two. The results are based upon results using graph size 5000 x 15000 and parameter Set C. When compared to the results obtained by the Genetic Algorithm and Tabu Search, the results offered by the Simulated Annealing appear less robust. While both the Genetic Algorithm and Tabu Search algorithm offer both decreasing levels of performance as the size of the graph increases, the number of very poor solutions where the ONVGR is equal to or less than 0.5 is greater for the Simulated Annealing than the

Tabu Search or Genetic Algorithm. The previous chapter, section 5.3, provided a rational for the discrepancy.

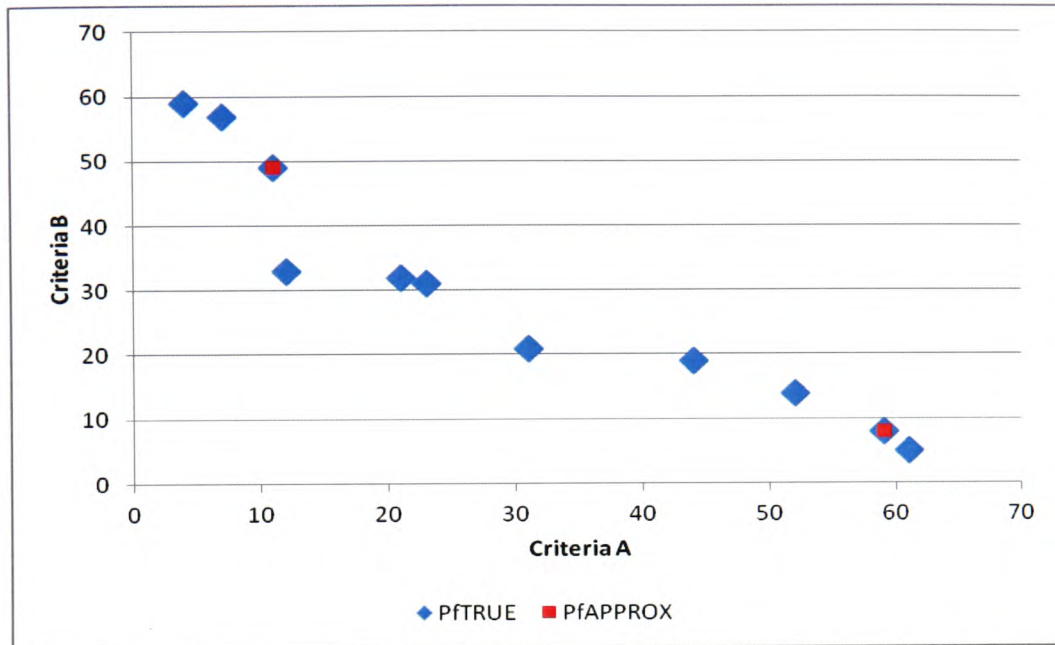


Figure 6.22 Comparison of *PfTRUE* and *PfAPPROX* From Simulated Annealing

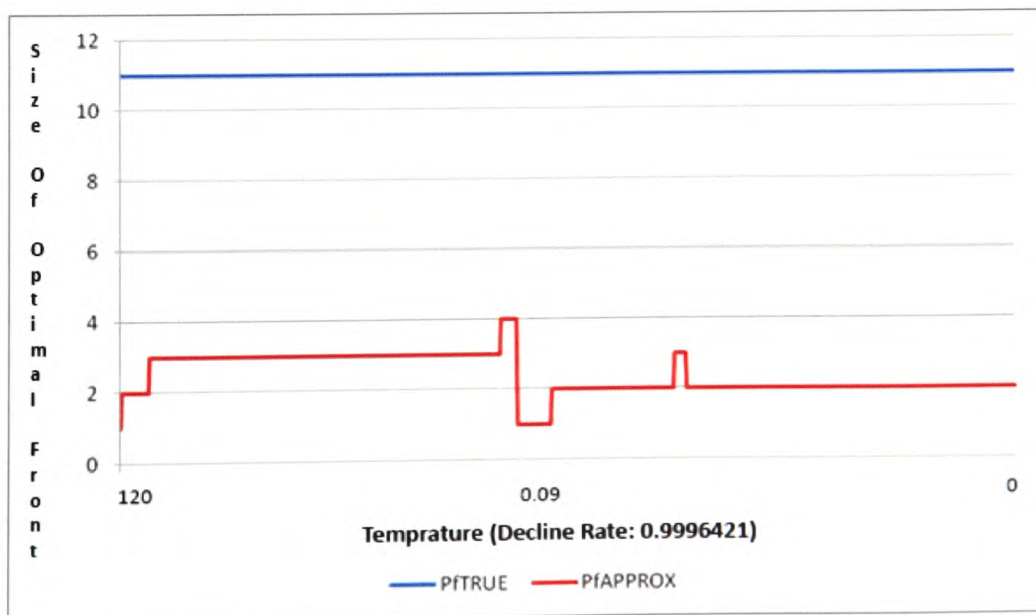


Figure 6.23 Size of *PfAPPROX* Over SA Processing Run

Range	Instances	$D(PfAPPROX; PfTRUE) \geq 2$	Solutions Found
3-5	43	3	3 (4.0)
6-8	36	7	4 (7)
8+	21	20	5 (10)

Table 6.4 *PfTRUE* Counts Against Acquired Using Simulated Annealing

6.3.4. Analysis of the PAES Algorithm

In this section, the results obtained using the PAES are discussed. For the purposes of the experimental phase the following number of iterations were selected:

- 20000 Iterations (Set A)
- 30000 Iterations (Set B)
- 40000 Iterations (Set C)
- 50000 Iterations (Set D)
- 60000 Iterations (Set E)

The performance of the algorithm is not studied in detail for the following reasons: primarily the algorithm is seen to behave in a similar way to the Simulated Annealing algorithm, with similar levels of performance offered as suggested in Figure 6.24. A review of the general behaviour of the algorithm, as it runs through the series of iterations required, demonstrates no major difference in behaviour. Secondly, as has been shown earlier, during the quantitative analysis phase, the major reason for the application of the algorithm, the crowding mechanism has little impact on the performance of the algorithm. Referring to Figure 6.24 the number of solutions in *PfTRUE* is limited when compared to the theoretical works in the literature. The lower

number of solutions in the front $PfTRUE$ highlights less clustering in the graphs reviewed, and therefore the crowding methodology has little impact.

6.3.5. Observations Regarding The Algorithms

Having completed the basic analysis of the algorithms the following observations are made regarding their performance. The overriding impact of the analysis is that the series of algorithms that have some form of ‘memory’ are able to provide a comparatively superior set of approximations of $PfTRUE$ than those without the concept of memory. The Genetic Algorithm approach and the Tabu Search both include this concept of memory, in the form of a population and Tabu list respectively. The memory concept allows the algorithms to explore more of the search space and therefore leads to an increase in the quality of the solutions provided. The PAES and Simulated Annealing methodology forego this concept of memory and are able to solve the MSPP with only a limited degree of success. The nature of the graph-based structure at the heart of the MSPP degrades the nature of the PAES and Simulated Annealing heuristics to something approaching a brute force technique, where the problem is decomposed to, in effect, a large number of random walks across the graph.

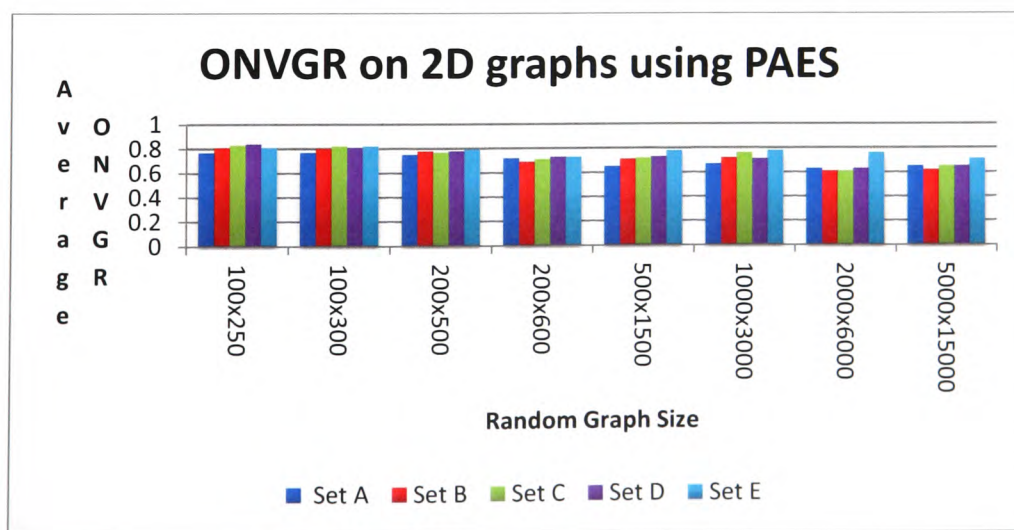


Figure 6.24 ONVGR on 2D Graphs Using PAES

6.4. Limitations in Experiments Undertaken

In this section the limitations of the experiments undertaken are discussed. The primary limitation in the experimental work concerns the selection of parameters to use in each of the heuristic algorithms developed as part of the study. It has been previously highlighted in the thesis that the original aim of the experimentation was to generate parameters that would result in generally equal runtimes for each of the heuristics. This however proved impossible. The starting point for the parameter selection was the selection of parameters for the Genetic Algorithm approach. These are similar and in certain cases identical to those used in Mooney (2004). In the case of the K-Geodesic approach care was taken to prevent the mechanism becoming a simple extension of an exhaustive search. It is plausible to suggest that the approach was too cautious. Given the runtimes seen it is possible that higher parameter values, such as tripling or quadrupling the number generations and population size may have been selected that would have given more positive results in terms of quality whilst maintaining a runtime comparable to the random walk technique. A similar limitation can be seen in the parameters selected for the remaining techniques (PAES, Simulated Annealing and Tabu Search). The parameters selected for each of those algorithms are able to complete a processing run more quickly than the random walk based genetic algorithm on smaller graphs. As the size of the graph increases however, the performance advantage of the three other heuristics quickly dissipates. The view of Geman and Geman (1984) is also considered. Geman and Geman highlight that given enough computational time the heuristics employed would have been able to generate a more complete approximation of the front $PfTRUE$. However adjusting the parameters would have either increased the runtime of the methods *or* resulted in a lower quality outcome for the experiments. Therefore whilst accepting that the parameter selection is open to criticism it is believed that the parameters selected are valid when runtime and general result quality are considered.

In the work undertaken no attempt has been made to apply early terminating scenarios to any of the algorithms implemented. This decision was made consciously and for what is believed to be a good reason. Early analysis of the 1+1 set of algorithms revealed there to be no discernible pattern as to when in a processing run the algorithm will acquire the final result set. In several instances a high number of iterations can be seen with no change in the set *PfAPPROX* before seeing a sudden burst of activity. The application of terminating conditions to those algorithms would it is believed have a negative impact on the overall quality of the results. For this reason that terminating conditions have not been applied.

A potential limitation can be seen in Algorithm 4.17. The limitation does not apply to this work given the nature of the test data but would perhaps need to be considered if alternative datasets were being used. The distance metric used does not normalize the values for each criterion. For instance in one criteria the range of possible values may be $\{0.01...1\}$, in another criteria $\{0.01...1,000,000\}$. As currently performed the algorithm may produce an inaccurate metric for the actual distance. However as previously stated this does not apply in this work.

The remaining possible limitation in the work can be seen in the concept of neighbourhood selection. Each of the three 1+1 algorithms requires a move to a related path in the neighbourhood of the solution in order to increase the coverage of the search space. In relation to this work the concept of neighbourhood would mean any valid path between the source and destination vertices. It is possible that the neighbourhood move selected is too large and could be further refined. However, the introduction of a procedure to identify additional members of the neighbourhood would lead to a substantial increase in the number of paths generated. Given that analysis of the algorithms has indicated that candidate path generation occupies the majority of the runtimes of the algorithms, increasing the size of neighbourhood selection would have a substantial negative effect of on the runtime. Alizamir *et al* (2009) highlight that computational complexity must play an important role in the selection of an appropriate

neighbourhood. This is in addition to the ability of the selected neighbourhood to cover search space.

6.5. Chapter Summary

The aim of this chapter was to review the heuristic approaches in terms of the quality of solutions provided. The 1+1 series of algorithms seen in the Tabu Search, Simulated Annealing and PAES approaches are certainly novel, with no existing literature regarding these methods application to the MSPP being seen in the literature.

The outcome of the experiments reveals that the Genetic Algorithm when applied with the random walk path generation technique being able to return the more complete set of optimal results. The remaining algorithms degrade more significantly and rapidly in the completeness of the approximations of the front $PfTRUE$. It should be noted however that in many cases the remaining alternative heuristics are still able to return a high quality approximation set of optimal solutions with no noticeable pattern of distance to $PfTRUE$ being seen when compared with the Genetic Algorithm approach. In cases where only a single solution is missing ($|PfAPPROX| = |PfTRUE| - 1$) then the heuristic algorithms are comparable.

Chapter 7: Conclusions and Future Work

7. Conclusions and Future Work

This thesis has investigated the feasibility of using a range of heuristic algorithms in order to solve the multi criteria shortest path problem. The current chapter provides a brief overview of the ways the aims of the work have been achieved before introducing the key outcomes. A series of possible areas for future work are then discussed. The chapter ends with a series of closing remarks about the work undertaken.

7.1. Research Methodology

An analysis of existing work into path planning was undertaken; this involved the review of a number of factors, including the psychological aspects of the path planning process. The key outcome of the review can perhaps be condensed into the following statement: *“There is no single optimal path for all users or situations”*. The nature of the individual and the approaches to the path planning process they undertake, together with the criteria they include in or exclude from the process ensures that there can be no single optimal solution. Historical approaches to the solution of single criterion shortest path problems were also considered with particular importance being placed on the application of data structures to increasing the performance of algorithms such as the Dijkstra shortest path algorithm. Various methods for achieving multi objective optimisation were considered including the works of authors such as Coello-Coello (2000) and Deb (2001). Metrics which have been developed as a means of measuring the effectiveness of the various approaches to the process of multi objective optimisation were also reviewed in this section of the work. Here the work of authors such as Zitzler *et al* (1999) and Veldhuizen (1999) was highlighted.

The work then progressed onto the specific area of the research topic, namely the application of the various heuristics previously reviewed to the problem of the multi

objective shortest path problem (MSPP). Algorithmic methods used to assist in the solution of the MSPP were reviewed in the form of techniques such as the Skriver and Anderson (2000) algorithm and the Martins and Climaco (1982) approach. The thesis then introduced the algorithms used in the study together with the test data selected. During the experimental work for the thesis Dijkstra's shortest path algorithm was extended to optimise several criteria. Existing heuristically driven techniques used to solve the MSPP were considered together with other heuristic techniques such as Simulated Annealing, the Tabu Search and PAES. Where the problem is not condensed into a single objective issue the Genetic Algorithm and ant colony optimisation dominate heuristics used to solve the MSPP. A recent exception can be seen in Liu *et al* (2012a) who apply a Simulated Annealing approach. In that regard the techniques developed in the thesis are certainly novel.

7.2. Summary of Research Outcomes

The solution of the MSPP is one that has been the subject of only a comparatively limited amount of scrutiny in the literature with research activity being sporadic and more often than not limited to algorithmic methods to solving the problem. Where AI based methodologies have been considered the primary mechanisms have been the Genetic Algorithm or more recently ant colony optimisation. It is worth perhaps reiterating the aims of the research. The three aims of the work are numbered for navigation purposes only with no importance implied in that ordering. General observations regarding each aim are then made.

AIM 1: To develop alternative heuristic techniques (to the Genetic Algorithm) for the solution of the MSPP

AIM 2: Assess the ability of those heuristic techniques to solve the MSPP against real world and synthetic graphs

AIM 3: Compare the alternative heuristic approach with algorithmic methods for the solution of the MSPP

7.2.1. Research Aim 1

The principle aim of the project was the investigation of AI based heuristics to assist in the solution of the MSPP. Chapter two of this work highlights that shortest path based applications where multiple criteria are considered are rare when view against single criteria approaches which continue to gain widespread research attention. The literature regarding solutions for the MSPP was considered. Existing methods for the solution of MSPP problems were identified in the form of Genetic Algorithms and more recent ‘nature’ based optimization methods such as ant colony optimization and invasive weed optimization. Only a single piece (Liu *et al*, 2012a) of work has been identified where other traditional methods of optimization such the Tabu Search and Simulated Annealing are used. As Liu *et al* (2012a) highlight, the scarcity of methods for the solution of the MSPP using those alternative heuristic methods is notable given the prevalence of the those methods in other routing problems such as the travelling salesman or associated routing problems such as pickup and delivery scheduling. Reviewing the literature on those two applications however quickly identifies that in the overwhelming majority of cases the solution to the introduction of additional criteria is solved with the use of aggregation and weighting effectively reducing the problem from one that is multi criteria in nature to one that is a relatively simple extension of a single criteria problem. The definition of appropriate weightings for each criterion is often far from simple.

Beyond the MSPP, literature regarding the application of multiple criteria to the heuristics of Simulated Annealing and the Tabu Search are rare and work which attempt model the multi criteria problem beyond that of simple extensions of single criteria are both rarer still and relatively recent. Notable example of extensions of the traditional approach can be seen in the works of Smith *et al* (2008) and Bandyopadhyay *et al* (2008) which attempt to make an assessment of the quality of a multi objective solution based upon some vector value between the fitness values of a solution and the current estimate of the Pareto optimal front. The general approach taken by Smith *et al* and Bandyopadhyay *et al* has been extended in this work to assist in the solution of the MSPP. In addition the vector difference technique has been applied to the Tabu Search approach. Finally an approach to the MSPP has been implemented based upon the PAES of Knowles and Corne (1999) has been implemented. Due to the scarcity of the work found using the techniques of Simulated Annealing and the Tabu Search the alternatives developed for this work are certainly to be considered novel. The novel nature of the work is considered to hold true despite the development of recent work (Liu *et al*, 2012a) for this problem. Several key differences such as path representation and mutation operators ensure both that the work of Liu *et al* and this work can be considered novel. The first of the three research aims has been met with the development of the Simulated Annealing and Tabu Search approaches. The methods associated with random walking for the Genetic Algorithm also lead to improvements in the solution of MSPP problems such as a degree of stability in terms of run times not seen in other works.

7.2.2. Research Aim 2

The second of the three aims of the undertaken research concerned the assessment of the techniques introduced in the previous section against a range of datasets. The developed heuristic algorithms have been tested against the performance of an implementation of a Genetic Algorithm approach to the solution of the MSPP. The performance of each of the heuristic approaches and the Genetic Algorithm approach were measured against a series of real world and synthetic graphs. The real world

datasets were acquired from Ordnance Survey datasets representing sections of the road network of the United Kingdom and range in size from 275 X 560 for the smallest of the graphs through to 58,583 X 123,248 for the largest. The synthetic datasets were generated using the SPRAND, a commonly used shortest path generator. The synthetic datasets ranged in size from 100 X 150 to 12,000 X 36,000.

The Genetic Algorithm using both a random candidate path generation method (random walking) and an algorithmic method in the form of a variation of the K shortest path algorithm are each able to generate complete or almost complete sets of the Pareto optimal front at lower graph sizes. Each of the other heuristics used are less able to return a complete set of optimal solutions regardless of the size of the graph being subjected to analysis. However, where $|PfAPPROX| \geq (|PfTRUE|-1)$ is considered a viable acceptance threshold then the general performance of the alternative heuristic approaches is seen to be much more positive where the quality of the approximated sets is largely equal regardless of the heuristic used.

The developed alternatives to the Genetic Algorithm are able to complete a processing run more quickly when considering smaller sized graphs. The speed differential is also true of the Genetic Algorithm with random walking on medium sized graphs with the heuristic approaches containing to operate more quickly. Both the heuristic approaches and Genetic Algorithm with random walking fall behind the performance of the Genetic Algorithm with K-Geodesic path generation on small to medium sized graphs. On larger sized graphs the performance of the Genetic Algorithm approach over the alternative heuristics continues to grow. As part of the research two promising walk mechanisms have been developed. The application of the procedures leads to decreased runtimes across graphs than can be seen elsewhere in the literature.

7.2.3. Research Aim 3

The third and final of the principle research aims involved the comparison between algorithmic methods for the solution of the MSPP and the heuristics developed for analysis during the course of this study. Three algorithmic methods for the performance of the MSPP have been implemented and tested for the study undertaken. The methods implemented include the performance of multiple runs of the Dijkstra with each run being used to analysis an individual criteria. Other methods include the exact methods of Climaco and Martins and Skriver and Andersen. Both the methods of Climaco and Martins and Skriver and Andersen will return the complete Pareto optimal set of paths for point-to-point queries.

The application of multiple runs of Dijkstra's shortest path algorithm returns only a limited subset of Pareto optimal solution, notably the extreme endpoints of the Pareto front or the shortest path for each criteria. In many cases however, such a limited representation of the front may be of interest particularly on real world networks where additional information sets can be supplied visually in the form of geographical information visualised as maps. The method has the benefits in that it is scalable both in terms of the size of the graph being considered and the number of criteria. During the testing the algorithm has been seen to perform in a runtime of two milliseconds on a graph sized 100 X 200 before increasing to 354 milliseconds on a graph sized 10000 X 20000 and when optimizing four criterion. The method outperforms all other techniques, whether algorithmic or heuristic based. The method has a worse case runtime of $\max_D C$ where D is the runtime of the Dijkstra shortest path algorithm and C is the number of criteria.

As highlighted in Chapter 5, the exact algorithmic methods of the Skriver and Andersen are Climaco and Martins outperform the average runtime seen in the heuristic methods on the smallest graph sizes. However, the average time seen in the heuristics does not reveal the extreme difference seen between the techniques. The PAES

algorithm for instance performs an analysis on smaller graphs (100 X 200) in an average runtime of 1.4 seconds compared to slightly more than 14 seconds for the Genetic Algorithm using random walking. The Skriver and Andersen method completes a processing run in 1.6 seconds. In terms of runtime the Skriver and Andersen operates close to that of the PAES (1.4 seconds), Simulated Annealing (1.7 seconds) and the Genetic Algorithm with K-Geodesics (1.8 seconds) approaches. The Skriver and Andersen technique however returns a more complete set of solutions. The Climaco and Martins approach may perform more quickly than the method of Skriver and Andersen or the heuristic methods. Like the Skriver and Andersen approach the method will return a complete set of optimal solution. There is however no methodology that can be employed prior to the analysis if faster runtimes (compared to the heuristics or Skriver and Andersen method) may be achievable. As the size of the graphs being subjected to analysis increases the performance advantage offered by the Skriver and Andersen over the heuristic approaches quickly dissipates. On a graph sized 200 X 400 (double the size of the 100 X 200 graph previously cited in this section) the Skriver and Andersen algorithm completes in a runtime of 11.6 seconds, an increase of over 700%. Amongst the heuristic methods employed in the study the highest increase in runtime seen in the runtimes is that of the PAES which increase from 1.4 seconds to 2.2 seconds between the graph sized 100 X 200 and 200 X 400, an increase of 57%. The Genetic algorithm sees the lowest increase from the heuristic approaches increasing in less than 1% (14.22 and 14.35).

The runtimes seen of exact methods of solving the MSPP over increasing graph sizes quickly demonstrate the value of approximate methods demonstrated in the form of the various heuristic techniques considered here. Of the considered heuristics the Genetic Algorithm shows a higher runtime over smaller graph sizes but also demonstrates a degree of scalability not seen in other heuristic methods or exact algorithms in terms of graph size. The heuristic approaches, together with the multiple runs of Dijkstra shortest path algorithm are not limited to any number of criteria where the two other algorithms of Skriver and Andersen and Climaco and Martins are limited to the solution of bi-criterion problems.

7.3. General Observations

This section offers a series of general observations regarding the research. These observations cover three areas: firstly, the importance of the concept of ‘memory’ is highlighted; secondly, a discussion on the scalability of the solutions is introduced. This relates to both the number of criteria and the size of the front. Finally, the methodology for the production of *PfTRUE* is reviewed.

7.3.1. Observations Regarding the use of ‘Memory’

The Genetic Algorithm and Tabu Search make use of memory structures in the form of the population and Tabu list. The series of algorithms that make use of some form of ‘memory’ outperform those solutions where no such structure is in place. The memory structure encourages the algorithm to seek out new areas to search for potential optimal paths. For the MSPP this is vital. Where it is absent then the heuristic functions will continue to admit the same path repeatedly and introduce a restriction of the search space examined leading to sub optimal solutions being returned.

7.3.2. Scalability

When applied to real world graphs the various attempts to solve the MSPP demonstrates poor scalability with respect to the graph size. This is due to the relative sparseness of real world graphs. It should be noted that the same condition applies to the algorithmic methods considered, including the approaches of Skriver and Andersen (2000) and Climaco and Martins (1982). The K-geodesic approach demonstrates an arguably acceptable level of scalability with a 300% increase between the smallest and largest graphs of London. The difference between the smallest and largest London graph using the random walk is around 5900%. However, the same is not true when applied to random graphs where much better levels of scalability can be seen with the

runtime random walk only increasing by 35% between the graph 100 x 200 and 12000 x 24000. This is despite the increase in the size of the graph being greater for the random graphs than the real world graph of London. In a later section future work has been suggested which may improve the scalability of the techniques to real world networks by reducing the sparseness of the graphs.

The MSPP when applied to real world graphs is practically demonstrated to be intractable. The graphs employed in the study undertaken are comparatively small, with a maximum size of fifty five thousand vertices and one hundred and twenty thousand edges. Table 7.1 presents the run times of some larger, countrywide example graphs. The table gives the vertex and edge count together with the source of the data. The runtimes achieved, taken together with the comparatively small sizes of the graphs employed, and the size increase to national level graphs require the introduction of the distinction between online and offline processing (Borodin and El-Yaniv, 1998). The runtimes shown demonstrate that online multi objective path planning is not from an end user perspective feasible for national level graphs. The problem however is feasible for large scale, off line processing. A limited subset of optimal paths can be developed quickly using extensions of single criteria approaches as demonstrated during the experimental work in this thesis.

Graph	Vertices	Edges	Source
UK	1,725,434	4,108,888	OS (Meriden)
Belgium	1,441,295	3,099,940	OSM
Netherlands	2,216,688	4,882,476	OSM
US	23,947,347	58,333,344	TIGER

Table 7.1 National Level Graph Sizes

7.3.3. Development of *PfTRUE* on Real World Graphs

This work resulted in the development of a robust methodology for the discovery of the front *PfTRUE* on real world graphs. Geodesic path length has been demonstrated to be a useful tool when calculating *PfTRUE* on random graphs. However, on high scale real world graphs the geodesic value may not be an accurate measure of the real world distance between two locations (represented as vertices). An alternative method based upon the Dijkstra shortest path algorithm and K shortest path algorithm has been developed which enables a robust measure of *PfTRUE* to be acquired on real world graphs, regardless of the scale of the road network data.

7.4. Future Work

During the undertaking of this study several related, yet separate topics, arose. This section briefly highlights areas of future research related to the topic.

7.4.1. Graph Generalisation

During the experimental phase of this study care has been taken to ensure that the underlying graph structure is not amended in any way. Nor does it require any specialised pre-processing of the graph data structure. The results of the experimental work performed raise the question as to whether or not this approach is practically correct. Chapter Four of the thesis introduced the notion of road features where a section of road is split into sections starting and terminating at when some given condition has been met. One avenue of future research is the investigation of methods that maintain the integrity of the underlying topology whilst at the same time decreases the number of vertices and edges making up the graph resulting in a decrease in the sparseness of the graph and resulting in the speeding up of the optimisation process. To demonstrate one possible example Figure 7.1 is presented. It presents an extreme

example of a road feature of the length d meters and consisting of seven edges and eight vertices.

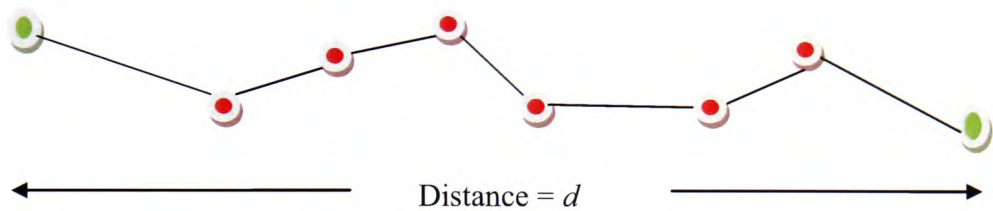


Figure 7.1 View of a Simple Uncondensed Road Segment

The future research envisaged would perform an investigation into the feasibility of reducing the network to a similar edge given in Figure 7.2, whilst ensuring that criteria such as length, road type and travel time etc are maintained. An initial review of the Shape File technical specification (ESRI, 1988) indicates such work would be feasible for real world networks. For a single feature, such as the example provided, the segment size is reduced from seven edges and eight vertices to a single edge with two vertices.

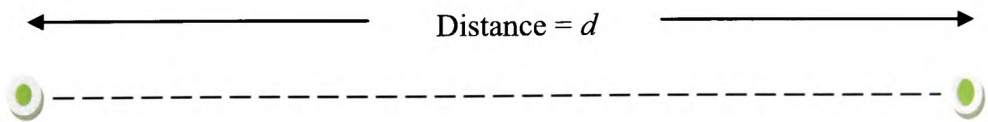


Figure 7.2 View of a Simple Condensed Road Segment

7.4.2. Analysis of Selection Methods

The central aim of the selection operator in evolutionary computation is to give preference to better individuals (those that are nearer to the solution) by allowing them to pass on their genes to the next generation and prohibit the entrance of worst fit individuals into the next generations. Reviewing the existing Genetic Algorithm methods for the solution of the MSPP demonstrates the dominance of a binary selection method. However, a wide range of alternative selection methods have been used elsewhere in the field of evolutionary computation. Alternative methods can be seen in the form of roulette wheel selection where solutions are selected according to their 'fitness'. A variation on roulette wheel selection has been used elsewhere in multi objective analysis and can be seen the form of Pareto Ranking where the fitness of a solution of considered as the ranking that the method would occupy during an iteration of Pareto extraction. For the MSPP however binary selection dominates the literature. An interesting avenue of future would possibly be introduction and examination of alternative methods of selection specifically for the MSPP.

7.4.3. Hybrid Algorithms

The application of the K shortest path algorithm to the generation of the candidate paths for the Genetic Algorithm is arguably an example of the development of a hybrid approach to the solution of the MSPP with the cross over properties of the Genetic Algorithm utilized in conjunction with the faster path generation techniques of the K shortest path. A number of additional steps that may increase the performance of the techniques considered in this work have been identified such as the use of seed values generated using algorithmic methods such as the Dijkstra shortest path algorithm or K shortest path using the Yen algorithm. The use of either would allow the algorithms to determine the extreme points of the Pareto front. In addition the use of the Yen algorithm for generating $K=2$ shortest paths should allow for the generation of a partially large number of paths related to the shortest path. In the process of acquiring

the second shortest path a large number of paths are generated but may be discarded. There may be some value in keeping these paths to check if they are of value. The model used by Yen ensures that there should be little additional overhead in generating the second shortest over generating just the shortest path.

Finally, a variation of the Climaco and Martins algorithmic method is proposed. An investigation is suggested where an upper limit on the maximum K value is set upon which the algorithm is terminated. The generation of the predefined K value of paths in each criterion may produce a high quality approximation of the optimal set. Given the high runtimes seen in certain tests on the heuristics of this study, the method certainly warrants further investigation.

7.5. Closing Comments

This work provides several original contributions in the field of multi-objective optimisation. Under limited circumstances traditional algorithmic methods, such as those presented by Martins and Climaco approach, can outperform all variations of the heuristic used in this study. The process is however entirely dependent on vertex selections and is certainly not universal. The evolutionary approach in the form of Genetic Algorithms is able to provide a general runtime much lower than that seen by the algorithmic approach of either Skriver and Andersen or Climaco and Martins, in certain cases, providing a high quality approximation of $PfTRUE$ in 50% of the runtime of algorithmic methods.

The heuristic and evolutionary approach to the MSPP is in essence a graph exploration issue. The solution of the MSPP requires the analysis of a large number of unique paths between two vertices. Where only limited exploration is possible the solution presented to the MSPP is poor. This is entirely logical.

This study has reviewed the feasibility of using heuristic based approaches to the multi objective shortest path problem. Results of the study show the heuristics presented, particularly the Genetic Algorithm approaches, present a good alternative in finding a subset of optimal solutions to the MSPP.

References

- Abraham, I., Delling, D., Goldberg, A. and Werneck, R. (2010) 'Alternative Routes in Road Networks'. *Proceedings of the 9th International Symposium on Experimental Algorithms (SEA'10)*. Ischia Island, Naples, Italy, May 20-22, 2010. pp. 23 – 34. Lecture Notes in Computer Science 6049, Springer-Verlang, ISBN: 9783642131929
- Abraham, I., Delling, D., Goldberg, A. and Werneck, R. (2011) 'A Hub-Based Labeling Algorithm for Shortest Paths on Road Networks'. *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA'11)*. Kolimpari, Chania, Crete, Greece, May 5-7, 2011. pp. 230 - 241. Lecture Notes in Computer Science 6630, Springer, ISBN: 9783642206610.
- Abraham, I., Delling, D., Fiat A., Goldberg, A. and Werneck, R. (2012) 'HLDB: Location-Based Services in Databases'. *Proceedings of the 20th ACM International Conference on Advances in Geographical Information Systems (GIS 2012)*. Redondo Beach, California, USA, November 6 - 9 2012. ACM, *In press*.
- Acuña D. and Parada V. (2010) 'People Efficiently Explore the Solution Space of the Computationally Intractable Traveling Salesman Problem to Find Near-Optimal Tours'. *PLoS One* 5(7), e11685. pp. 1 - 10. Accessed: 10/10/2012. URL: <http://www.plosone.org/article/info%3Adoi/10.1371/journal.pone.0011685>
- Agarwal, R., Caesar, M., Godfrey, P., Zhao. B. (2012) 'Shortest Paths in Less than a Millisecond'. *Proceedings of the 2012 ACM workshop on Workshop on online social networks (WOSN '12)*. Helsinki, Finland — August 13 - 17, 2012, USA. pp. 37 – 42. ACM, ISBN: 978145014800
- Ahn, C. and Ramakrishna, R. (2002) 'A Genetic Algorithm for Shortest Path Routing Problem and the Sizing of Populations'. *IEEE Transactions on Evolutionary Computation* 6 (6). pp. 566 – 579.
- Ahuja, R., Magnanti, T. and Orlin, J. (1993). *Network Flows: Theory Algorithms and Applications*. Prentice Hall Inc. ISBN: 013617549X
- Aittokallio, T. and Schwikowski. B., (2006) 'Graph-Based Methods for Analysing Networks in Cell Biology'. *Briefings in Bioinformatics* 7 (3). pp. 243 - 255.

Alahakoon, T., Tripathi, R., Kourtellis, N., Simha, R. and Iamnitchi, A. (2011) 'K-Path Centrality: a New Centrality Measure in Social Networks'. *Proceedings of the 4th Workshop on Social Network Systems (SNS '11)*. Salzburg, Austria, April 10-13 2011. Article 1. ACM, ISBN: 978145030728

Albert, R. and Barabasi A. (2002) 'Statistical Mechanics of Complex Networks'. *Review of Modern Physics*. 74 (1). pp. 47-97.

Aldious D. and Fill J. (1999) *Reversible Markov Chains and Random Walks on Graphs*. Online: <http://www.stat.berkeley.edu/~aldous/RWG/book.html> Accessed: 19/9/2012

Alizamir, S., Rebennack, S. and Pardalos, P. (2009) 'Improving the Neighborhood Selection Strategy in Simulated Annealing using the Optimal Stopping Problem'. *Global Optimisation: Focus on Simulated Annealing*. pp. 363 – 382.

Alon, N., Avin, C., Koucky, M., Kozma, G., Lotker, Z. and Tuttle, M. (2008) 'Many Random Walks are Faster than One'. In *Proceedings of the 20th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA 2008)*, Munich, Germany. June 14-16 2008. pp. 119–128. ACM, ISBN 9781595939739.

Apple (2012) 'Maps:Maps Take A Whole New Turn.' Online: <http://www.apple.com/ios/ios6/maps/>. Retrived: 14-6-2012.

Applegate, D. L., Bixby, R. M., Chvátal, V. and Cook, W. J. (2007). *The Traveling Salesman Problem*. Princeton University Press. ISBN 0691129932.

Arentze, T., Feng, T., Robbroeks, F., van Brakel, M. and Huibers, R. (2012) 'Compliance with and Influence of a new in-car Navigation System for Trucks: Results of a Field Test'. *Transport Policy* 23, pp. 42 - 49.

Auger. A., Bader J., Brockhoff D. and Zitzler E. (2012) 'Hypervolume-based Multiobjective Optimisation: Theoretical Foundations and Practical Implications'. *Theoretical Computer Science*. 435. pp. 75 - 103.

Austin, T., Fagen, R., Penney, W. and Riordan. J. (1959) 'The Number of Components in Random Linear Graphs'. *Annals of Mathematical Statistics*. 30 (3). pp. 747 – 754.

Avin, C. and Brito, C. (2004) 'Efficient and Robust Query Processing in Dynamic Environments Using Random Walk Techniques'. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*. Berkeley, California, USA, April 26-27, 2004. pp. 277 – 286. IEEE, ISBN 1581138466.

Azaria, A., Rabinovich, Z., Kraus, S., Goldman, C. and Tsimhoni O. 2012 'Giving advice to people in path selection problems'. *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '12)*. Valencia, Spain, June 4-8 2012. pp. 459 - 466. ACM, ISBN: 0981738117

Bach , B., Spritzer, A., Lutton, E. and Fekete, J. (2012) 'Interactive Random Graph Generation with Evolutionary Algorithms'. *Proceedings of 20th International Symposium on Graph Drawing*. September 19-21, 2012. Redmond, Washington, USA. *In Press*.

Back, T. (1996). 'Evolutionary Algorithms in Theory and Practice: Evolutionary Strategies, Evolutionary Programming and Genetic Algorithms.' *Oxford University Press*. ISBN 0195099710

Bader, J. and Zitzler. E. (2008) 'HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimisation'. TIK Report 286, Computer Engineering and Networks Laboratory (TIK), ETH Zurich.

Bagheri, N., Benwell, G. and Holt, A. (2005) 'Measuring Spatial Accessibility to Primary Health Care.' *The 17th Annual Colloquium of the Spatial Information Research Centre*. University of Otago, Dunedin, New Zealand. URL: <http://hdl.handle.net/10523/756>. Accessed: 10/11/2012

Baker, J. (1987) 'Reducing Bias and Inefficiency in the Selection Algorithm.' *Proceedings of the 2nd international conference of generic algorithms and their applications*. Massachusetts Institute of Technology Cambridge, Massachusetts, USA, July 28-31, 1987. pp 14 - 22. Psychology Press, ISBN: 0805801596

Bailenson, J.N., Shum, M.S. and Uttal, D.H. (1998). 'Road Climbing: Principles Governing Asymmetric Route Choices on Maps'. *Journal of Environmental Psychology* 18. pp. 251 – 264.

Balboa, J. and López, F. (2008) 'Generalization-oriented Road Line Classification by Means of an Artificial Neural Network'. *GeoInformatica* 12 (3). pp. 289 - 312.

Bandyopadhyay, S.; Saha, S., Maulik, U. and Deb, K. (2008) 'A Simulated Annealing-Based Multiobjective Optimisation Algorithm: AMOSA'. *IEEE Transactions on Evolutionary Computation*. 12 (3). pp. 269 - 283.

Baños R., Ortega, J., Gil, C., Fernández, A. and de Toro, F., (2012) 'A Simulated Annealing-Based Parallel Multi-Objective Approach to Vehicle Routing Problems with Time Windows'. *Expert Systems with Applications*. ISSN 0957-4174.

Barabasi, A. and Albert. R. (1999) 'Emergence of Scaling In Random Networks'. *Science* 286. pp. 509 – 512.

Baran, B. and Schaerer, M. (2003) 'A Multiobjective Ant Colony System for Vehicle Routing Problem with Time Windows'. *IASTED International Multi-Conference on Applied Informatics*. Innsbruck, Austria, February 10-13, 2003, IASTAD Press, ISBN: 0889863458

Barrat, A. and Weigt, M. (2000) 'On the properties of small-world network models', *European Physics Journal B: Condensed Matter*. 13 (1). 547 – 560.

Bast, H., Funke, S., and Matijevic. F. (2006) Ultrafast Shortest-Path Queries via Transit Nodes. *Proceedings of the Ninth DIMACS Implementation Challenge*. Piscataway, New Jersey, US, November 13-16, 2006. pp. 175 – 192. American Mathematical Society, ISBN: 9780821843833.

Bast, H., Carlsson E., Eigenwillig A., Geisberger, A., Harrelson, C., Raychev, V. and Viger, F. (2010) 'Fast Routing in Very Large Public Transportation Networks using Transfer Patterns'. *Proceedings of the 18th annual European conference on Algorithms: Part I (ESA'10)*. Liverpool, United Kingdom, September 6–8, 2010. pp.290 - 301. ACM, ISBN: 3642157807

Batagelj, V. and Brandes, U. (2005) 'Efficient Generation of Large Random Networks'. *Physical Review E*. 71 (3). Article: 36113.

Bauer, R. and Delling. D. (2010) 'SHARC: Fast and Robust Unidirectional Routing'. *Journal of Experimental Algorithmics* 14 (4). pp 13-26.

Bayati, M., Han Kim, J. and Saberi. A. (2007). 'A Sequential Algorithm for Generating Random Graphs'. *Proceedings of the 10th International Workshop on Approximation and the 11th International Workshop on Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX '07/RANDOM '07)*. Princeton, NJ, USA, August 20-22, 2007. pp. 326 - 340. Lecture Notes in Computer Science 4627. Springer-Verlang, ISBN: 9783540742074

Bellman, R. (1958) 'On a Routing Problem'. 'Quarterly of Applied Mathematics' 16. pp. 87 – 90.

Berge, C. (1973) *Graphs and Hypergraphs*. Elsevier. ISBN: 0720424534

Bekhor, S., Ben-Akiva, M.E. and Ramming, M.S. (2006). 'Evaluation of Choice Set Generation Algorithms for Route Choice Models'. *Annals of Operations Research*. 144 (1). pp. 235 – 247.

Berg-Insight (2012), Mobile Navigation Services and Devices, LBS Research Series. URL: <http://www.berginsight.com/ReportPDF/Summary/bi-lbsseries2012-sum.pdf>. Accessed: 28/9/2012

Beume, N., Fonseca, C., López-Ibáñez, M. and Jan V. (2009) 'On the Complexity of Computing the Hypervolume Indicator'. *IEEE Transactions on Evolutionary Computation*. 13 (5). pp. 1075 - 1082.

Bezerra, L., Goldbarg, E., Goldbarg, M. and Buriol L. (2011) 'GRACE: A Generational Randomized ACO for the Multi-Objective Shortest Path Problem'. *Proceedings of the 6th International Conference on Evolutionary Computation (EMO'11)*. Ouro Preto, Brazil, April 5-8, 2011. pp. 535 – 549. Springer, ISBN: 9783642198922.

Bezerra, L., Goldbarg, E., Goldbarg, M. and Buriol, L. (2013). 'Analyzing the Impact of MOACO Components: An Algorithmic Study on the Multi-objective Shortest Path Problem'. *Expert Systems with Applications*. 40 (1). pp. 345 - 355.

Bonsall, P. (1992) 'The Influence of Route Guidance on Route Choice in Urban Networks'. *Transportation*. 19 (1). pp. 1 - 23.

Boost (2012) *The Boost C++ libraries*. www.boost.org, Accessed 1st February, 2012.

Borodin, A. and El-Yaniv, R. (1998) '*Online Computation and Competitive Analysis*'. Cambridge University Press. ISBN: 0-521-56392-5.

Brabham, D. (2008) 'Crowd Sourcing as a Model for Problem Solving An Introduction and Cases'. *Convergence* 14 (1). pp.75 - 90.

Brandao, J. and Eglese, R. (2008) A deterministic tabu search algorithm for the capacitated arc routing problem. *Computers & Operations Research* 2008. 35(4) pp. 1112 – 26.

Brandão, J and Mercer, A. (2012) '*A tabu search algorithm for the multi-trip vehicle routing and scheduling problem*'. *European Journal of Operational Research*, 100 (1). pp. 180 – 191.

Brander. A. W and Sinclair. M C. (1995) '*A Comparative Study of k-Shortest Path Algorithms*' *Proceedings of the 11th UK Performance Engineering Workshop*. Liverpool, United Kingdom, September 5-6 1995, Springer, ISBN 0354760083.

Brockhoff D., Friedrich. T., Hebbinghaus, N., Klein C., Neumann F. and Zitzler E. (2007) '*Do Additional Objectives Make a Problem Harder?*'. *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO '07)*. London, United Kingdom, July 7-11 2007. pp. 765 - 772. ACM, ISBN 9781595936974.

Brockhoff, D., and Zitzler, E. (2006a) '*Are All Objectives Necessary? On Dimensionality Reduction in Evolutionary Multiobjective Optimisation*'. *Proceedings of Parallel Problem Solving from Nature 2006*, Reykjavik, Iceland, September 9-13 2006. pp. 533 – 542. Lecture Notes in Computer Science 4193, Springer, ISBN 3540389903.

Brockhoff, D. and Zitzler, E. (2006b) '*Dimensionality Reduction in Multiobjective Optimisation: The Minimum Objective Subset Problem*'. *Proceedings of Operations Research 2006*. Selected Papers of the Annual International Conference of the German Operations Research Society (GOR), Jointly Organized with the Austrian Society of Operations Research (ÖGOR) and the Swiss Society of Operations Research (SVOR), Karlsruhe, Germany, September 6-8, 2006. 2007. Springer, ISBN 9783540699941

Brumbaugh-Smith J. and Shier D. (1989) '*An Empirical Investigation of Some Bicriterion-Shortest Path Algorithms*'. *European Journal of Operational Research*. 43 (2). pp. 216 - 224.

Burgess, R. and Darken. J. (2004) '*Realistic Human Path Planning using Fluid Simulation*'. *Proceedings of Behaviour Representation in Modelling and Simulation (BRIMS) 2004*. Arlington, Virginia, US, May 17-20, 2004.

Brummit, B. (2007). '*The road to better path-finding*'. Google Official Blog. URL: (<http://googleblog.blogspot.co.uk/2007/11/road-to-better-path-finding.html>) Accessed: 10/10/2012

Car, A. (1997) '*Hierarchical Spatial Reasoning: Theoretical Consideration and Its Application to Modelling Way finding*'. Technical Report. Department Of Geoinformation, Technical University of Vienna: 195.

Carlyle, W. and Wood, R. (2005) '*Near-Shortest and K-Shortest Simple Paths*'. *Networks*. 46 (2). pp. 98 – 109.

Cardiff (2007) Empowering the public with accurate real-time information, **url no longer available**.

Chakraborty, B., Madedda, T. and Chakraborty, G. (2005) 'Multi Objective Route Selection for Car Navigation System using Genetic Algorithm'. *Proceedings of the 2005 IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications*. Espoo, Finland, June 28-30 2005. pp. 190 - 195. IEEE, ISBN: 0780389425.

Cheikh M., Jarboui B. and Loukil T. (2010) 'A Genetic Algorithms to Solve the Bicriteria Shortest Path Problem'. *Electronic Notes in Discrete Mathematics*. 36 (1). pp. 851 - 858.

Chen, Y. (1994) 'Finding the K Quickest Simple Paths in a Network'. *Information Processing Letters* 50. pp. 89 - 92.

Chen, H., Ding, Zhaohui. W., Tong. Y, Lavanya. D. and Chen, J (2009) 'Semantic Web for Integrated Network Analysis in Biomedicine'. *Briefings in Bioinformatics*. 10 (2). pp. 177-192.

Chen, Y. W., Wang, C. H. and Lin, S. J. (2006) 'A Multi-Objective Geographic Information System for Route Selection of Nuclear Waste Transport'. *Omega*. 36 (3). pp. 363 – 372.

Cherkassky, B. Goldberg, A. and Radzik, T. (1994) 'Shortest Paths Algorithms: Theory and Experimental Evaluation'. *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '94)*. Arlington, Virginia, USA, January 23-25, 1994. pp. 516 - 525. ACM, ISBN: 0898713293

Cherkassky, B., Goldberg, A. and Radzik, T. (1996) 'Shortest Paths Algorithms: Theory and Experimental Evaluation'. *Math Programming*. 73 (2). pp. 129 – 174.

Chitra, C. and Subbaraj, P. (2010) 'A Nondominated Sorting Genetic Algorithm for Shortest Path Routing Problem'. *International Journal of Electrical and Computer Engineering*. 39 (1). pp. 55 - 63.

Chitra, C. and Subbaraj, P., (2012). 'A Nondominated Sorting Genetic Algorithm Solution for Shortest Path Routing Problem in Computer Networks. *Expert Systems with Applications*. 39 (1). pp. 1518 - 1525.

Chorus, G. C., Molin, E. J. E. and Van Wee, B. (2006) 'Use and Effects of Advanced Traveller Information Services (ATIS): a review of the literature'. *Transport Reviews*. 26 (2). pp. 127 - 149.

Christensen, J., Marks, J. and Shieber, S. (1995) 'An Empirical Study of Algorithms For Point-Feature Label Placement'. *ACM Transactions on Graphics*. 14(3). pp 203 – 232.

Climaco, J. and Martins E. (1982) 'A Bicriterion Shortest Path Algorithm'. *European Journal of Operational Research*. 11 (4). pp. 399 - 404.

Coello-Coello, C. (2000) 'An Updated Survey of GA Based Multi objective Optimisation Techniques'. *ACM Computing Surveys (CSUR)*. 32 (2). pp. 109 - 143

Coello-Coello, C. and Salazar-Lechuga, M. (2002) 'MOPSO: A Proposal for Multiple Objective Particle Swarm Optimisation' *IEEE Congress on Evolutionary Computation (CEC'2002)*, Honolulu, Hawaii, US, May 17-21, 2002. pp. 1051 - 1056. IEEE, ISBN: 0780372786

Coello-Coello, C and Lamont, G. (2005) '*Applications of Multi-Objective Evolutionary Algorithms*'. World Scientific Publishing. ISBN: 9789812561060.

Cohen, J. L. (1978) '*Multiobjective Programming and Planning*'. Academic Press, New York, USA. ISBN: 9780486432632.

Cordeau, J. and Maischberger, M. (2012) A parallel iterated tabu search heuristic for vehicle routing problems. *Computers & Operations Research*. 39 (3). pp. 2033 – 2050.

Cormen, T., Leiserson, C. and Rivest, R. (2001) '*Introduction to Algorithms*'. The MIT Press, Cambridge, Massachusetts, USA. ISBN: 9780072970548.

Corne, D. W., Deb, K., Fleming, P. J. and Knowles, J. D. (2003) 'The Good of the Many Outweighs the Good of the One: Evolutionary Multi-Objective Optimisation'. *Connections: The Newsletter of the IEEE Neural Networks Society*. 1 (1) pp. 9 – 13.

Costelloe, D., Mooney, P., and Winstanley, A. C. 'From Random Walks to Pareto Optimal Paths'. *Proceedings of the 12th Irish Conference on Artificial Intelligence and Cognitive Science*, Maynooth, Ireland, September 5-7 2001. pp. 309 - 318.

Coutinho-Rodrigues, J. M., Climaco, J.C.N. (1994) 'A PC-Based Interactive Decision Support System for Two Objective Direct Delivery Problems'. *Journal of Business Support*. 15 (1). pp. 305 - 322.

Coutinho-Rodrigues, J. M., Climaco, J. C. N. & Current, J. R. (1999) 'An Interactive Biobjective Shortest Path Approach: Searching for Unsupported Nondominated Solutions'. *Computers and Operations Research*. 26 (8). pp. 789 – 798.

- Coutinho-Rodrigues, J., Tralhão, L. and Alçada-Almeida, L. (2012) 'Solving a Location-Routing Problem with a Multiobjective Approach: the Design of Urban Evacuation Plans'. *Journal of Transport Geography*. 22. pp. 206 - 218.
- Crichigno, J. and Baran, B. (2004) 'Multiobjective Multicast Routing Algorithm for Traffic Engineering' *Proceedings of the 13th International Conference on Computer Communications and Networks (ICCCN 2004)*. Chicago, Illinois, USA, October 11-13 2004. pp. 301 - 306. IEEE, ISBN: 0780388143.
- Current, J., Min. H. and Schilling, D. (1990) 'Multiobjective Analysis of Facility Location Decisions' *European Journal of Operational Research*. 49 (3). pp 295 - 307.
- Current, J. and Marsh, M. (1993) 'Multi Objective Transportation Network Design and Routing Problems - Taxonomy and Annotation'. *European Journal of Operational Research*. 65 (3) pp 1 - 15.
- Czyzak, P. and Jaskiewicz, C. (1998) 'Pareto Simulated Annealing – a Metaheuristic Technique for Multipleobjective Combinatorial Optimisation'. *Journal of Multi-Criteria Decision Analysis*. 7 (1). pp 34 – 47.
- Das Sarma, A., Gollapudi, S., Najork, M., Panigrahy, R. (2010) 'A Sketch-Based Distance Oracle for Web-Scale Graphs'. *Proceedings of the 3rd ACM international conference on Web search and data mining (WSDM '10)*. New York City, NY, USA, February 03 - 06, 2010. pp. 401 – 410. ACM, ISBN: 9781605588896
- Davies, C. and Lingras, P. (2003). 'Genetic Algorithms for Rerouting Shortest Paths in Dynamic and Stochastic Networks'. *European Journal of Operational Research*. 144 (1). pp. 23 - 38.
- Davoodi, M., Panahi, F., Mohades, A. and Hashemi, S. (2013). 'Multi-Objective Path Planning in Discrete Space'. *Applied Soft Computing*. 13 (1). pp. 709 – 720.
- Deb, K. (1998) *Limitations of evolutionary computation methods*. In: Back, T., Fogel, D. and Michalewicz, Z. (eds.), *Handbook of Evolutionary Algorithms*, Institute of Physics Publishing Ltd. ISBN: 0750303921
- Deb, K. (1999) *Evolutionary Algorithms for Multi-Criterion Optimisation in Engineering Design*. In: Miettinen, K., Makela, M. M., Neittaanmaki, P. and Periaux, J. (eds.), *Evolutionary Algorithms in Engineering and Computer Science*, Wiley, ISBN: 9780471999027
- Deb, K. (2001) *Multi-Objective Optimisation using Evolutionary Algorithms*. John Wiley & Sons, Chichester, United Kingdom. ISBN: 9780471873396.

Deb, K., Pratap, A., Agarwal S. and Meyarivan, T. (2002) 'A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II'. *IEEE Transactions on Evolutionary Computation*. 6 (2). pp. 182 – 197.

Dechter, R. and Pearl. J. (1985) 'Generalized Best-First Search Strategies and the Optimality of A*'. *Journal Of The ACM*. 32 (3). pp. 505 - 536.

De Meo, P., Ferrara, E., Fiumara, G. and Ricciardello, A. (2012) 'A Novel Measure of Edge Centrality in Social Networks'. *Knowledge-Based Systems*. 30. pp. 136 – 150.

Delling, D. and Wagner, D. (2007) 'Landmark-Based Routing in Dynamic Graphs'. *Proceedings of the 6th International Workshop on Experimental Algorithms (WEA07)*. Rome, Italy, June 6-8, 2007. pp. 52 - 65. Lecture Notes in Computer Science 4525, Springer, ISBN 9783540728443

Delling, D. and Wagner, D. (2009) 'Pareto Paths with SHARC'. *Proceedings of the 8th International Symposium on Experimental Algorithms (SEA '09)*. Dortmund, Germany, June 3-6, 2009. pp. 125 – 136. Lecture Notes in Computer Science, 5526, Springer-Verlang, ISBN: 9783642020100

Delling, D., Goldberg, A., Razenshteyn, I. and Werneck, R. (2010) 'Graph Partitioning with Natural Cuts'. Microsoft Research Technical Report Number: MSR-TR-2010-164

URL: <http://research.microsoft.com/apps/pubs/default.aspx?id=142349> Accessed: 10/11/2012

Delling, D., Goldberg, A., Nowatzyk, A. and Werneck, R. (2011a) 'PHAST: Hardware-Accelerated Shortest Path Trees', *Proceedings of the 25th International Parallel and Distributed Processing Symposium (IPDPS'11)*, Anchorage, Alaska, USA, 16-20 May, 2011. pp. 921 - 931. IEEE, ISBN: 9781612843728

Delling, D., Goldberg, A., Pajor, T. and Werneck, R. (2011b) 'Customizable Route Planning'. *Proceedings of the 10th International Symposium on Experimental Algorithms (SEA'11)*. Kolimpari, Chania, Crete, Greece, May 5-7, 2011. pp. 376 – 387. Lecture Notes in Computer Science 6049, Springer, ISBN: 9783642206610

Dell'Olmo, P., Gentili, M. and Scozzari, A. (2005) 'On Finding Dissimilar Pareto-Optimal Paths'. *European Journal of Operational Research*. 162 (1). pp.70 - 81.

Dial, R., Glover. F., Karney, D. and Klingman, D. (1979) 'A Computational Analysis of Alternative Algorithms and Labelling Techniques for finding shortest path trees'. *Networks* 9. pp. 215 - 248.

Dick, R., Rhodes, D. and Wolf, W. (1998) 'TGFF: Task Graphs For Free'. *Proceedings of the 6th International Workshop on Hardware/Software Codesign*. pp. 97-101. Washington, DC, USA, March 15-18 1998. IEEE, ISBN: 0818684429.

Diestel, R. (2005). '*Graph Theory*'. *Graduate Texts in Mathematics Volume 173*. Springer-Verlag. pp. 6 – 9. ISBN: 9783642142789.

Dijkstra, E. (1959) 'A Note on Two Problems In Connection With Graphs'. *Numeriche Mathematik* 1. pp. 269 - 271.

Doerr, B., Happ, E., and Klein, C. (2008) 'Crossover can Provably be Useful in Evolutionary Computation'. *Proceedings of the 2008 Conference on Genetic and Evolutionary Computation Conference (GECCO)*. Atlanta, GA, USA — July 12 - 16, 2008. pp. 539 – 546, ACM, ISBN: 9781605581309

Dorigo, M. (1992) '*Optimisation, Learning and Natural Algorithms*'. PhD thesis, Politecnico di Milano, Italie.

Dorigo, M. and Socha, K. (2006). '*An Introduction to Ant Colony Optimization*'. Technical report, TR/IRIDIA/2006-010 IRIDIA, Université Libre de Bruxelles.

Dolev, S., Schiller, E. and Welch, J.L. (2002) 'Random Walk for Self-Stabilizing Group Communication in Ad Hoc Networks'. *IEEE Transactions on Mobile Computing*. 5 (7). pp. 893 - 905.

Duckham, M. and Kulik, L. (2003) 'Simplest Paths: Automated Route Selection for Navigation'. *Spatial Information Theory: Foundations Of Geographic Information Science*. pp.169 - 185. Lecture Notes in Computer Science 2825, Springer-Verlang, ISBN: 9783540201489.

Duckham, M., Winter, S., and Robinson, M. (2010). 'Including landmarks in routing instructions'. *Journal of Locatation Based Services* 4 (1). pp. 28-52.

Efentakis, A., Pfoser, D. and Agnès Voisard. (2011) 'Efficient Data Management in Support of Shortest-Path Computation'. *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Computational Transportation Science (CTS '11)*. Chicago, Illinois, USA, November 1-4, 2011. pp. 28 - 33. ACM, ISBN: 9781450310345.

Elsässer, E. and Sauerwald T. (2011) 'Tight Bounds for the Cover Time of Multiple Random Walks'. *Theoretical Computer Science*. 414 (24). pp. 2623 - 2641.

Eppstein, D. (1997) 'Finding the K Shortest Paths'. Technical Report. University Of California, Irvine, California, US.

Erdos, P. and Renyi, A. (1959) 'On random graphs I'. *Publicationes Mathematicae* 6. pp. 290 – 297.

Escobar, J., Linfati, R. and Toth, P. (2013) 'A two-phase hybrid heuristic algorithm for the capacitated location-routing problem' *Computers & Operations Research*. 40 (1) 70 - 79.

ESRI, (1988) ESRI Shapefile Technical Description. URL: <http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf> . Accessed 1/1/2012.

ESRI (2005) 'Hierarchal Routes in ArcGIS Network Analysis: An ESRI White Paper' URL: http://downloads.esri.com/support/whitepapers/other_/ArcGIS_NA_Hierarchical_Routes_Aug05.pdf Accessed: 5/12/2012.

Fan, H., Hua, Z., Li, J. and Yuan, D. (2004) 'Solving a Shortest Path Problem by Ant Algorithm' *Proceedings of 2004 International Conference on Machine Learning and Cybernetics*. Shanghai, China, August 26-29 2004. pp. 3174 – 3177. IEEE, ISBN: 0780384032

Fan, L. and Mumford, C. (2010). 'A Metaheuristic Approach to the Urban Transit Routing Problem'. *Journal of Heuristics*. 16 (3). pp. 353 - 372.

Fang, Z., Zong, X., Li, Q. and Li, Q. (2011). 'Hierarchical Multi-Objective Evacuation Routing in Stadium Using Ant Colony Optimization Approach' . *Journal of Transport Geography*. 19 (3). pp. 443 - 451.

Fieldsend, J., Everson, R. and Singh, S. (2001) 'Extensions to the Strength Pareto Evolutionary Algorithm'. *IEEE Transactions on Evolutionary Computation*. Submitted but superseded by: Fieldsend, J., Everson, R. and Singh, S. (2003) 'Using unconstrained elite archives for multi-objective optimisation'. *IEEE Transactions on Evolutionary Computing*. 7 (3). pp. 305 – 323.

Flickr (2012) 'The App Garden'. URL: <http://www.flickr.com/services/api/> Accessed: 11/10/2012

Flores, R., Koster, M., Lindner, I. and Molina, E. (2012) 'Networks and collective action'. *Social Networks*. 34 (4). pp. 570 – 584.

Floyd, R. (1962). 'Algorithm 97: Shortest Path'. *Communications of the ACM*. 5 (6). pp. 345

Fogel, J. L., Owens, A. and Walsh, M. (1966) *Artificial Intelligence through Simulated Evolution*. John Wiley & Sons. ISBN: 047133250X.

Fonseca, C. and Fleming, J. (1993) 'Genetic Algorithms for Multi Objective Optimisation: Formulation, Discussion and Generalization'. *Proceedings of the 5th International Conference on Genetic Algorithms*. Urbana-Champaign, IL, USA, June 1993. pp. 416 - 423. Morgan-Kaufmann, ISBN: 1558602992

Fonseca C, Paquete L and Lopez-Ibez (2006) 'An Improved Dimension-Sweep Algorithm for the Hypervolume Indicator'. In: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*. Vancouver, British Columbia, Canada. July 16-21, 2006. pp. 1157 - 1163. IEEE, ISBN: 0780394879

Forbes. (2012) 'Facebook's 1 Billion Users: Why The Sky Is Still The Limit'. URL: <http://www.forbes.com/sites/haydnshaughnessy/2012/10/04/facebook-1-billion-users-why-the-sky-is-still-the-limit/> Accessed: 11/10/2012

Forrest, S (1993) 'Genetic Algorithms: Principles of Natural Selection Applied to Computation'. *Science*. 261. pp. 872 - 878.

Ford. L. and Fulkerson, D. (1962) *Flows in Networks*. Princeton University Press. ISBN: 9780691146676

Fowler, J., Dawes, C. and Christakis. N. (2009) Model of genetic variation in human social networks. *Proceedings of the National Academy of Sciences*. 106 (6). pp. 1720 – 1724.

Fox. B. (1975) 'K-th Shortest Paths and Applications to the Probabilistic Networks' *Operations Research Society Of America/TIMS Joint National Management*. 23 (1). Article: B263

Fredman, M., and Tarjan, R. (1987) 'Fibonacci Heaps and Their Use in Improved Network Optimisation Algorithms'. *Journal of the Association for Computing Machinery*. 34 (4). pp. 596 - 615.

Furtado, V., Melo, A., Coelho, A., Menezes, R. and Perrone, R. (2009) 'A Bio-Inspired Crime Simulation Model' *Decision Support Systems*. 48 (1). pp. 282 - 292,

Garey, M. and Johnson, D. (1979) *Computers and Intractability: A Guide to the Theory*

Of NP-Completeness'. W. H. Freeman. ISBN: 9780716710455.

Gandibleux X., Mezdaoui, N. and Fravville, A. (1997) 'A Tabu Search Procedure to Solve Multi Objective Combinatorial Optimisation Problems'. *Proceedings of the Second International Conference on Multi-Objective Programming and Goal Programming*. Torremolinos, Spain. pp. 291 - 300. Lecture Notes in Economics and Mathematical Systems 445. Springer-Verlang. ISBN: 354061768X

Gandibleux, X. and Freville, A. (2000) 'Tabu Search Based Procedure for Solving the 0-1 Multiobjective Knapsack Problem: the two objectives case'. *Journal of Heuristics*. 6 (3). pp. 361 – 383.

Gen, M., Runwei C. and Dingwei, W. (1997) 'Genetic Algorithms for Solving Shortest Path Problems'. *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC'97)*. Indianapolis, Indiana, US, April 16 - 19, 1997. pp.401 - 406. IEEE, ISBN: 0780339509

Gen, M. and Lin, L. (2004) 'Multiobjective Hybrid Genetic Algorithm for Bicriteria Network Design Problem'. *The 8th Asia Pacific Symposium on Intelligent and Evolutionary Systems*. Cairns, Australia, December 6 - 7, 2004. Monash University, ISBN: 9780646443409

Gendreau, M., Hertz, A. and Laporte, G. (1992) 'New Insertion and Post-Optimisation Procedures for the Traveling Salesman Problem'. *Operations Research*. 40 (6). pp. 1086-1094.

Geisberger, R. (2008) '*Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks*'. Institut für Theoretische Informatik Universität Karlsruhe (TH), Germany. Unpublished Phd Thesis.

Geisberger, R., Sanders, P., Schultes, D., and Delling, D. (2008) 'Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks'. *Proceedings of the 7th Workshop on Experimental Algorithms (WEA'08)*. Provincetown, Cape Cod, Massachusetts, USA, May 30-2 June 2008. pp . 319 - 333. Lecture Notes in Computer Science 5038, Springer-Verlang, ISBN: 9783540685487

Geman, S., and Geman, D. (1984) 'Stochastic Relaxation, Gibbs' Distribution and the Bayesian Restoration of Images'. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 6 (6). pp. 721 - 741.

Ghanizadeh, A.; Sinaie, S.; Abarghouei, A.A. and Shamsuddin, S.M. (2010) 'A Fuzzy-Particle Swarm Optimisation Based Algorithm for Solving Shortest Path Problem'. *Proceedings Of The 2nd International Conference on Computer Engineering and Technology (ICCET)*. Chengdu, China, April 16-17, 2010. pp 404 - 408. IEEE, ISBN: 9781424463473.

Gilbert, E.N. (1959) 'Random Graphs' *Annals of Mathematical Statistics*. 30. pp. 1141 – 1144.

Gjoka, M., Kurant, M., Butts, C.T. and Markopoulou, A. (2010) 'Walking in Facebook: A Case Study of Unbiased Sampling of OSNs'. *Proceedings of the 29th Conference on Computer Communications (INFOCOM, 2010)*. San Diego, California, USA. March 15-19, 2010. pp. 1 – 9. IEEE, ISBN: 9781424458387

Gkantsidis, C. Mihail, M. and Saberi, A. (2004) 'Random Walks in Peer-To-Peer Networks'. *Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies. (INFOCOMM 2004)*. Hong Kong, March 7 - 11, 2004. IEEE, ISBN: 0780383559

Glover F. (1986) 'Future paths for Integer Programming and Links to Artificial Intelligence'. *Computers and Operations Research*. 5. pp. 533 - 549

Glover, F. (1989) 'Tabu Search: Part 1'. *Operations Research Society of America Journal*. 1 (3). pp. 190 - 206.

Glover, F. and Laguna M. (1998) '*Tabu Search: Background and Relevance*'. Springer-Verlang. ISBN: 0792381874

Goldberg, A., Radzik, T., (1993) 'A Heuristic Improvement of the Bellman-Ford Algorithm'. *Applied Mathematics Letters*. 6 (3). pp. 3 - 6.

Goldberg, A. and Silverstein, C. (1995) *Implementations of Dijkstra's Algorithm Based on Multi-Level Buckets*. Technical Report 95-187. NEC Research Institute, Inc.

Goldberg, A. and Harrelson, C. (2005) 'Computing The Shortest Path: A Search Meets Graph Theory'. *Proceedings of the 16th annual ACM-SIAM symposium on Discrete algorithms (SODA '05)*. Vancouver, British Columbia, Canada, January 23-25, 2005. pp. 156 - 165. SIAM, ISBN: 0898715857

Goldberg, A. (2011) "Shortest Paths in Road Networks". *School on Graph Theory, Algorithms and Applications*. Erice, Scilly, Italy, September 25 – October 3 2011.

Goldberg, D. and Richardson, J. (1987) 'Genetic Algorithms with Sharing for Multimodal Function Optimisation'. *Proceedings of the 2nd International Conference on Genetic Algorithms and their Applications*. Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, July 28-31, 1987. pp. 41 – 49. Psychology Press, ISBN: 0805801596
Goldberg, D. (1989) '*Genetic Algorithms in Search, Optimisation & Machine Learning*'. Addison Wesley. ISBN: 0201157675.

Goldreich, O. (2008) *Computational Complexity: a Conceptual Perspective*, Cambridge University Press. ISBN: 9780521884730

Goldstein, L. and Waterman, M. (1988) 'Neighbourhood Size in the Simulated Annealing Algorithm'. *American Journal of Mathematical and Management Sciences*. 8 (34) pp. 409 – 423.

Golledge, R. (1995) 'Path Selection and Route Preference in Human Navigation: a Progress Report' *Spatial Information Theory: A Theoretical Basis for GIS*. pp. 207 - 222. Lecture Notes in Computer Science 988. Springer-Verlang, ISBN: 9783540603924

Golledge, R. and Stimson, R. (1997) 'Decision Making and Choice Behaviour Models'. *Spatial behaviour: A geographic perspective*. pp. 31 – 70. Guilford Press, ISBN: 9781572300507

Google Inc. (2012) 'The Encoded Polyline Algorithm'. URL: <https://developers.google.com/maps/documentation/utilities/polylinealgorithm> Accessed: 4/10/2012

Gräbener, T., Berro, A. and Duthen, Y. (2010) 'Time Dependent multiobjective Best Path for Multimodal Urban Routing'. *Electronic Notes in Discrete Mathematics*. 36 (1) pp. 487 - 494.

Granat, J. and Guerriero, F. (2003) *The Interactive Analysis of the Multi Criteria Shortest Path Problem by the Reference Point Method*. European Journal of Operational Research. 151 (1). pp. 103 - 118.

Grandinetti, L., Guerriero, F., Laganá, D. and Pisacane, O. (2012) 'An Optimization-Based Heuristic for the Multi-objective Undirected Capacitated Arc Routing Problem. *Computers & Operations Research*'. 39 (12) pp. 2300 - 2309

Gubichev, A., Bedathur, S., Seufert, S. and Weikum, G. (2010) 'Fast and Accurate Estimation of Shortest Paths in Large Graphs'. *Proceedings of the 19th ACM international conference on Information and knowledge management (CIKM '10)*. Toronto, Ontario, Canada, October 26-30, 2010. pp. 499 - 508. ACM, ISBN: 9781450300995

Häckel, S., Fischer, M., Zechel, D. and Teich T. (2008) 'A Multi-Objective Ant Colony Approach for Pareto-Optimization Using Dynamic Programming'. *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2008)*. Atlanta, Georgia, USA, July 12-16, 2008. pp. 33 – 40. ACM, ISBN: 9781605581309

Haigh, K., Veloso, M. (1995) 'Route Planning by Analogy'. *Proceedings of the 1st International Conference on Case-Based Reasoning Research and Development*. Sesimbra, Portugal. October 23 - 26, 1995. Lecture Notes in Computer Science 1010, Springer-Verlang, ISBN: 9783540605980

Hallam, C. Harrison, K.J. and Ward, J. (2001) 'A Multiobjective Optimal Path Algorithm'. *Digital Signal Processing*. 11 (2) pp. 133 – 143.

Hamed, A. (2010) 'A Genetic Algorithm for Finding the K Shortest Paths in a Network'. *Egyptian Informatics Journal*, 11 (2) pp.75 - 79.

Hansen, M. P. and Jaskiewicz, A. (1998) *Evaluating the Quality of Approximations of the Non-Dominated Set*. Technical Report. Institute of Mathematical Modeling, Technical University of Denmark. IMM-REP-1998-7.

Hansen, P. (1979) 'Bicriterion Path Problems'. *Multiple Criteria Decision Making: Theory and Applications*. Lecture Notes in Economics and Mathematical Systems 177, Springer. pp. 109 – 27.

Hansen, P. (1997) 'Tabu Search for Multi Objective Optimisation: MOTS'. *Proceedings of the 13th International Conference on Multiple Criteria Decision Making*. Cape Town, South Africa, January 1997. Lecture Notes in Economics and Mathematical Systems 465, Springer-Verlang, ISBN: 9783540647416

Hapke, M., Jaskiewicz, A. and Slowinski R. (2000) 'Pareto Simulated Annealing for Fuzzy Multi-Objective Combinatorial Optimisation'. *Journal of Heuristics*. 6 (3). pp. 329–345.

Harris Interactive (2007). *National Study Shows GPS Adoption Rates Relatively Low, but Offers Recommendations to Accelerate Market Penetration* URL:
<http://www.harrisinteractive.com/NEWS/allnewsbydate.asp?NewsID=1241>.
Retreived:28/09/2012

Hart, P., Nilson, N., Raphael, B. (1968) 'A Formal Basis for the Heuristic Determination of Minimum Cost Path'. *IEEE Transactions on Science System and Cybernetics*. 4 (2). pp. 100-107.

Hart, W., Krasnogor, N., and Smith, J., editors (2004) 'Recent Advances in Memetic Algorithms'. *Studies in Fuzziness in and Soft Computing* 166. Springer-Verlang, ISBN: 9783540229049.

- He, F., Qi, H. and Fan, Q. (2007) 'An evolutionary algorithm for the multi-objective shortest path problem'. *Proceedings of the 2007 International Conference on Intelligent Systems and Knowledge Engineering (ISKE 2007)*. October 15-16, 2007, Chengdu, China. Atlantis Press, ISBN: 9789078677048
- Hershberger, J. Maxel, M. and Suri, S. (2007) 'Finding the K Shortest Simple Paths: A New Algorithm and its Implementation'. *ACM Transactions on Algorithms*. 3 (4). Article 45.
- Hertz, A., Jaumard, B., Ibeiro, C.C. and Formosinho-Filho, W.P. (1994) 'A Multi-Criteria Tabu Search Approach to Cell Formation Problems in Group Technology with Multiple Objectives'. *Operations Research*. 28 (3). pp. 303 – 328.
- Hilger, M., Köhler, E., Möhring, R. and Schilling, H. (2006) 'Fast Point-to-Point Shortest Path Computations with Arc-Flags'. *Proceedings of the Ninth DIMACS Implementation Challenge*. Piscataway, New Jersey, US, November 13-16, 2006. pp 41 – 73. American Mathematical Society, ISBN: 9780821843833
- Hoffman, W. and Pavley, R. (1959) 'A Method for the Solution of the Nth Best Path Problem'. *Journal of the Association for Computing Machinery*. 6 (4). pp.506 – 514.
- Hoitkotter, J. and Beasley, D. (2000) 'The Hitchhikers Guide to Evolutionary Computation'. URL: <http://www.aip.de/~ast/EvolCompFAQ/> Accessed: 25/11/2012.
- Holland, J. H. (1975) 'Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence'. MIT Press. ISBN: 9780262581110.
- Holzer, M., Schulz, F., Wagner, D. and Willhalm, T. (2005) 'Combining Speed-Up Techniques for Shortest-Path Computations'. *Journal of Experimental Algorithmics*. 10. Article 2.5.
- Hopcroft, J., Motwani, R. and Ullman, J. (2007) 'Introduction to Automata Theory, Languages, and Computation'. Addison Wesley. ISBN: 9780321455369
- Horoba, C. (2010) 'Exploring the Runtime of an Evolutionary Algorithm for the Multi-Objective Shortest Path Problem'. *Journal of Evolutionary Computation*. 18 (3). pp. 357 - 381.
- Horn, J, Nafpliotis, N. and Goldberg, D. (1994) 'A Niche Pareto Genetic Algorithm For multi objective optimisation'. *Proceedings of the 1st IEEE Conference on Evolutionary Computation*. Orlando, Florida, USA, June 27-29, 1994. pp. 82 – 87. IEEE, ISBN: 0780318994
- Horn, J. (1996) 'Multicriteria Decision Making and Evolutionary Computation'. Technical Report, ILLiGAL Report No. 9600X, Department of Engineering, University of Illinois at Urbana- Champaign, Urbana, Illinois 61801, USA.

Hubschneider, M. (2012) 'Preferred Truck Routes Meet Navigation'. *Procedia - Social and Behavioral Sciences*. 39. pp. 490 – 494

Hung, K.T., Liu, J.S. Chang, Y.Z. (2007) 'A Comparative Study of Smooth Path Planning for a Mobile Robot by Evolutionary Multi-Objective Optimization'. *Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation*. Jacksonville, Florida, USA, June 20–23 2007. pp. 254-259. IEEE, ISBN: 1424407907

Ioannides. Y. (2006) '*Random graphs and social networks: An economics perspective*'. Technical Report 0518, Department of Economics, Tufts University, 2005.

Iredi, S., Merkle, D., and Middendorf M. (2001) 'Bi-Criterion Optimisation with Multi Colony Ant Algorithms'. *Proceedings of the 1st International Conference on Evolutionary Multi-Criterion Optimisation (EMO'01)*. Zurich, Switzerland, March 7-9, 2001. Lecture Notes in Computer Science 1993, Springer-Verlang, ISBN: 9783540417453

Jackson, P. (1999) '*Introduction to Expert Systems*'. Addison Wesley. ISBN: 9780201876864

Jacob, R., Marathe, M. and Nagel, K. (1999) 'A Computational Study of Routing Algorithms for Realistic Transportation Metworks'. *Journal of Experimental Algorithms*. 4 (6). pp. 167-178.

Jaeggi, D., Parks, G., Kipouros, T. and Clarkson, P. (2008) 'The Development of a Multi-Objective Tabu Search Algorithm for Continuous Optimisation Problems'. *European Journal of Operational Research*. 185 (3). pp. 1192-1212.

Jaffrès-Runser K.; Gorce, J. and Ubeda, S. (2008). 'Mono- and Multiobjective Formulations for the Indoor Wireless LAN Planning Problem'. *Computers & Operations Research*. 35 (12). pp. 3885-3901

Johnson, D. (1977) 'Efficient Algorithms for Shortest Paths in Sparse Networks' *Journal of the ACM*. 24 (1). pp. 1–13

Jones, D., Mirrazavi, K. and Tamiz, M. (2002) 'Multi-Objective Meta Heuristics: An Overview of the Current State-of-the-Art'. *European Journal of Operational Research*. 137 (1). pp. 1–9.

Kauer, H., Ye, T., Kalyanaraman, S. and Vastola, K. (2003) 'Minimizing Packet Loss by Optimizing OSPF Weights using Online Simulation'. *Proceedings of the 11th IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer Telecommunications Systems (MASCOTS 2003)*. Orlando, Florida, USA, October 12-15 2003. pp. 79-86. IEEE, ISBN: 0769520391.

Kennedy, J. and Eberhart R. (1995) 'Particle Swarm Optimisation'. *Proceedings of the IEEE International Conference on Neural Networks*. Perth, Australia, November 27-1 December 1995. pp. 1942-1948. IEEE, ISBN: 0780327683

Kolahan, F., Abolbashari, M.H. and Mohitzadeh, S. (2007) '*Simulated Annealing Application for Structural Optimisation*'. *Proceedings of the World Academy of Science, Engineering and Technology*. 26. p. 606-609.

Kellerer, H., Pferschy, U. and Pisinger, D. (2004) '*Knapsack Problems*'. Springer. ISBN: 9783540402862.

Kingston, J. (1998) '*Algorithms and Data Structures: Design, Correctness, Analysis*'. Addison-Wesley. ISBN: 9780201403749.

Kirkpatrick, S., Gelatt, C. and Vecchi, M. (1983) '*Optimization by Simulated Annealing*'. *Science*. 220 (4598). pp. 671 - 680.

Knowles, J., and Corne, D. (1999) 'The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Pareto Multi objective Optimisation'. *Proceedings of the 1999 Congress on Evolutionary Computation (CEC'99)*. Washington D.C, USA, July 6-9 1999. pp. 98 - 105. IEEE, ISBN: 0780355369

Krasnogor, N. (2002) *Studies on the Theory and Design Space of Memetic Algorithms*. Unpublished PhD Thesis. University of the West of England.

Krebs, V. (2001) 'Mapping networks of terrorists cells'. *Connections*. 24 (3). pp. 43 - 52.

Krishnamurthy, B. and Wills, C. (2009) 'On the Leakage of Personally Identifiable Information via Online Social Networks'. *Proceedings of the 2nd ACM workshop on Online social networks (WOSN '09)*. Barcelona, Spain — August 7 - 12, 2009. pp. 25 - 30. ACM, ISBN: 9781605584454

Kulturel-Konak, S., Smith, A. and Norman., B. (2006) 'Multi-objective Tabu Search using a Multinomial Probability Mass Function'. *European Journal of Operational Research*. 169 (3). pp. 918 - 931.

Lamont, G. (2005) 'Multi Objective Evolutionary Algorithms: What, Why, and Where (A tutorial)'. *Proceedings of Evolutionary Multi Objective Optimisation 2005 (EMO2005)*.

Guanajuato, Mexico, March 9-11, 2005. Lecture Notes in Computer Science 3010, Springer-Verlang, ISBN: 9783540249832.

Larranaga, P., Kuijpers, C., Murga, R., Inza, I. and Dizdarevic, S. (1999) 'Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators'. *Artificial Intelligence Review*. 1 (13). pp. 129 – 170.

Lawler, E. (1972) 'A Procedure for Computing the K Best Solutions to Discrete Optimisation Problems and its Application to the Shortest Path Problem'. *Management Science*. 18 (1). pp. 401 - 405.

Lawler, E. (1977) 'Comment on Computing the k Shortest Paths in a Graph'. *Communications of the ACM*. 20 (8). pp. 603 – 604.

Li, H., Guensler, R. and Ogle, J. (2005) 'Analysis of Morning Commute Route Choice Patterns Using Global Positioning System-Based Vehicle Activity Data'. *Transportation Research Record: Journal of the Transportation Research Board*. 1929. pp. 162–170.

Li, H. and Landa-Silva, D. (2011) 'An Adaptive Evolutionary Multi-Objective Approach based on Simulated Annealing'. *Journal of Evolutionary Computation*. 19 (4). pp. 561 – 595.

Li, M., Wang, W. and Zhang, Y. (2010) 'Research on driver experience based route planning method'. *Proceedings of the 2010 IEEE Intelligent Vehicles Symposium (IV)*. San Diego, California, USA, 21-24 June 2010. pp.78-82

Liu, B., Choo, S., Lok, S., Leong, S., Lee, S., Poon, F. and Tan, H. (1994) 'Integrating Case-Based Reasoning, Knowledge-based Approach and Dijkstras' Algorithm for Route Finding'. *Proceedings of the 10th International Conference on Artificial Intelligence for Applications*. San Antonio, Texas, USA. March 1-4 1994. pp. 149 - 155. IEEE, ISBN: 081865550X

Liu, B. (1996) 'Intelligent Route Finding: Combining Knowledge. Cases and an Efficient Search Algorithm'. *Proceedings of the 12th European Conference on Artificial Intelligence (ECAI96)*. Budapest, Hungary, August 12-16 1996. pp. 380-384. John Wiley and Sons, ISBN: 9780471968092.

Liu, C., Pai, T., Chen-tien, C. and Chang-Ming, H. (2001) 'Path-planning algorithms for public transportation systems'. *Proceedings of the 4th International IEEE Conference on Intelligent Transportation Systems*. Oakland, California, USA. August 25-29 2001. pp. 1061-1066. IEEE, ISBN: 0780371941

Liu, C. (2002) 'Best-path planning for public transportation systems'. *Proceedings Of the 5th IEEE International Conference on Intelligence Transport Systems*. Singapore. September 3-6 2002. pp. 834 – 839. IEEE, ISBN: 0780373898

Liu, L., Mu, H., Yang, X., He, R. and Li, Y. (2012a). 'An Oriented Spanning Tree Based Genetic Algorithm for Multi-Criteria Shortest Path Problems'. *Applied Soft Computing*. 12 (1). pp. 506 -515.

Liu, L., Mu, H., Luo, H. and Li, X. (2012b). 'A Simulated Annealing for Multi-Criteria Network Path Problems', *Computers & Operations Research*. 39 (12). pp. 3119 – 3135.

Luger, G. (2002) *'Artificial Intelligence: Structures and Strategies for Complex Problem Solving'*. Addison Wesley. ISBN: 9780321545893

Luxen, D. and Vetter, C. (2011) 'Real-time routing with OpenStreetMap data'. *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (GIS '11). Chicago, Illinois, USA — November 01 - 04, 2011. pp. 513 - 516. ACM, ISBN: 9781450310314

Lyons, G. (2006) 'The Role of Information in Decision-Making with Regard to Travel'. *Intelligent Transport Systems*. 153 (3). pp. 199-212.

Lyons, G., Bonsall, P. and Sampson, E. (2008) 'Human Behaviours to Moving People More Effectively'. ESRC Seminar Series: Mapping the Public Policy Landscape.

Machuca, E. and Mandow, L. (2011) 'Multiobjective Heuristic Search in Road Maps'. *Expert Systems with Applications*. 39 (7). pp. 6435-6445

Machuca, E., Mandow, L., Pérez de la Cruz J. and Ruiz-Sepulveda, A. (2012) 'A Comparison of Heuristic Best-First Algorithms for Bicriterion Shortest Path Problems'. *European Journal of Operational Research*. 217. (1). pp. 44-53.

Mahfoud, S. (1995) *'Niching Methods for Genetic Algorithms'*. Unpublished PhD Thesis. University Of Illinois.

Managbanag J., Witten T., Bonchev D, Fox L. and Tsuchiya M. (2008) 'Shortest-Path Network Analysis Is a Useful Approach toward Identifying Genetic Determinants of Longevity'. PLoS ONE. 3(11). Article: e3802.

URL:<http://www.plosone.org/article/metrics/info%3Adoi%2F10.1371%2Fjournal.pone.0003802;jsessionid=FAE750B0C327B7081AAC80408FF27ED4>

Accessed: 10/11/2012

Mandow, L. and Pérez de la Cruz, J.L. (2010) 'Multiobjective A* search with Consistent Heuristics'. *Journal of the ACM*. 57 (5). Article 27.

Martins, E. (1984) 'On a Multicriteria Shortest Path Problem'. *European Journal of Operational Research*. 16 (2). pp. 236-245.

Martins, E. and Santos, J. (1999). '*The Labeling Algorithm for the Multiobjective Shortest Path Problem*'. Departamento de Matematica, Universidade de Coimbra, Portugal, Technical Report: TR-99/005.

Mayer, M. (2011). '*Marissa Mayer Presents...*'. Online Resource, URL: <http://www.youtube.com/watch?v=TIwzbZq5uyY> Accessed 21/9/2012

McGlohon, M., Akoglu, L. and Faloutsos. C. (2008) 'Weighted Graphs and Disconnected Components: Patterns and a Generator'. *ACM Special Interest Group on Knowledge Discovery and Data Mining (KDD08)* Las Vegas, Nevada, USA, August 24 - 27, 2008. pp. 524-532. ACM, ISBN: 9781605581934

Melhorn, K and Sanders P. (2008) '*Algorithms and Data Structures: The Basic Toolbox*'. Springer. ISBN: 9783540779773

Merris, R. (2000) '*Graph Theory*'. Wiley Series in Discrete Mathematics and Optimisation. ISBN: 9780471389255.

Merz, P. (2000) '*Memetic Algorithms for Combinatorial Optimisation Problems: Fitness Landscapes and Effective Search Strategies*'. Unpublished PhD thesis, Department of Electrical Engineering and Computer Science, University of Siegen.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. and Teller, E. (1953) 'Equations of State Calculations by Fast Computing Machines'. *The Journal of Chemical Physics*. 21 (6). pp. 1087-1092.

Michalewicz, Z. and Michalewicz, M. (1997) 'Evolutionary Computation Techniques and their Applications'. *Proceedings of the IEEE International Conference on Intelligent Processing Systems*. Beijing, China, October 28-31 1997. pp 14-21. IEEE, ISBN: 0780342534.

Milgram, S. (1967) 'The Small World Problem' *Psychology Today* 1 (1). pp. 61 - 67.

Miller, B. and Goldberg, G. (1996) 'Genetic Algorithms, Selection Schemes, and the Varying Effects of Noise'. *Journal of Evolutionary Computation*. 4 (2). pp. 113-131.

Miller, D. and Star, M. (1967) 'The Structure of Human Decisions'. Prentice Hall. ISBN: 9780138546878.

Mislove, A., Koppula, H., Gummadi, K., Druschel, P. and Bhattacharjee, B. (2008) 'Growth of The Flickr Social Network'. *Proceedings of the 1st workshop on Online Social Networks (WOSN '08)*. Seattle, WA, USA — August 17 - 22, 2008. pp. 25 - 30. ACM, ISBN: 9781605581828

Mitchell, M. (1996) 'An Introduction to Genetic Algorithms'. The MIT Press. ISBN: 9780262631853

Mnuemoto, M., Takai, Y. and Sato, Y. (1998) 'A Migration Scheme for the Genetic Adaptive Routing Algorithm'. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*. San Diego, California, USA. October 11-14, 2008. pp. 2774-2780. IEEE, ISBN: 0780347781

Modesti, P. and Sciomachen, A. (1998) 'A Utility Measure for Finding Multiobjective Shortest Paths in Urban Multimodal Transportation Networks'. *European Journal of Operations Research*. 111 (3). p.495 – 508.

Möhring, R. H., Schilling, H., Schütz, B., Wagner, D., and Willhalm, T. (2005) 'Partitioning Graphs to Speed up Dijkstra's Algorithm'. *Proceedings of the 4th Workshop on Experimental Algorithms (WEA'05)*. Santorini Island, Greece, May 10-13, 2005. pp. 189--202. Lecture Notes in Computer Science 3503, Springer-Verlang, ISBN: 3540259201.

Möhring, R. H., Schilling, H., Schütz, B., Wagner, D., and Willhalm, T. (2007) 'Partitioning Graphs to Speedup Dijkstra's Algorithm'. *Journal of Experimental Algorithmics*. 11 (1). Article 2.8.

Mooney, P. (2004) '*Multi criteria Path Optimisation on Graphs*'. Unpublished Phd Thesis. Department of Computer Science, National University of Ireland, Maynooth.

Mooney, P., Winstanley, A. (2006) '*Evolutionary Algorithm for Multicriteria Path Optimization Problems*'. *International Journal of Geographical Information Science*. 20 (4). pp. 401 - 423.

Moscato, P. (1993) 'An Introduction to Population Approaches for Optimisation and Hierarchical Objective Functions: A Discussion on the Role of Tabu Search'. *Annals of Operations Research*. 41 (1). pp. 85 – 121.

Müller, J. (2010) 'Approximative Solutions to the Bicriterion Vehicle Routing Problem with Time Windows'. *European Journal of Operational Research*. 202 (1). pp. 223 - 231.

Nam, DK. and Park, CH. (2000) 'Multiobjective Simulated Annealing: a Comparative Study to Evolutionary Algorithms'. *International Journal of Fuzzy Systems*. 2 (2) p.87 – 97.

New York Time (2012) '*For Impatient Web Users, an Eye Blink Is Just Too Long to Wait*'. URL:http://www.nytimes.com/2012/03/01/technology/impatient-web-users-flee-slow-loading-sites.html?_r=1 Accessed:4/12/2012

Newman, M. (2001) 'The Structure of Scientific Collaboration Networks'. *Proceedings of the National Academy of Science*. 98 (2). pp. 404 – 409.

Newman. M, Watts. D. and Strogatz. S, (2002) 'Colloquium Paper: Random Graph Models of Social Networks'. *Proceedings of the National Academy of Science*. 99 (1). pp. 2566 – 2572.

Nijkamp P and Van Deft A. (1971) '*Multi-Criteria and Regional Decision Making*'. Martinus Nijhoff Science Division, Leiden. ISBN: 9789020706895.

Oduwaga, V., Tiwari, A. and Roy, R. (2005) 'Evolutionary Computing in Manufacturing Industry: an Overview of Recent Applications'. *Applied Softcomputing*. (5). pp. 281 - 299.

OS (2012), Technical Specification of the ITN layer, <http://www.ordnancesurvey.co.uk/oswebsite/support/products/os-mastermap/itn-layer-technical-specification/index.html> Accessed 1st Febuary, 2012.

Pacheco, J. and Mart, R. (2006) 'Tabu Search for a Multi-Objective Routing Problem'. *Journal of the Operational Research Society*. 57 (1). pp. 29 – 37.

Pallottino, S. and Scutella, M. G. (1997) '*Shortest Path Algorithms in Transportation Models: Classical and Innovative Aspects*'. Technical Report: TR-97-06, Dipartimento di Informatica, Universita di Pisa, Italy.

Pangilinan, J. and Janssens, G. (2007) 'Evolutionary Algorithms for the Multiobjective Shortest Path Problem'. *International Journal of Computer and Information Science and Engineering*. 1 (5). pp. 323 - 327.

Papinski, D., Scott, D. and Doherty, S. (2009). 'Exploring The Route Choice Decision-Making Process: A Comparison of Planned and Observed Routes Obtained using Person-based GPS'. *Transportation Research Part F: Traffic Psychology and Behaviour*. 12 (4). pp. 347 - 358.

Papinski, D. and Scott, D. (2011) 'A GIS-Based Toolkit for Route Choice Analysis'. *Journal of Transport Geography*. 19 (3). pp. 434 – 442.

Pascoal, M, and Martins. E. (2003) 'A New Implementation of Yen's Ranking Loopless Paths Algorithm'. *4OR – Quarterly Journal of the Belgian, French and Italian Operations Research Societies*. 1 (2). pp. 121 - 133.

Pendelton, C. (2012) 'Bing Maps New Routing Engine'. *Bing Map Community Blog*. URL: http://www.bing.com/community/site_blogs/b/maps/archive/2012/01/05/bing-maps-new-routing-engine.aspx. Accessed: 4/10/2012

Pennock, D. M., Flake, G. W., Lawrence, S., Glover, E. J. and Giles, C. (2002) 'Winners don't take all: Characterizing the competition for links on the web' *Proceedings of the National Academy of Science*. 99 (8). pp. 5207 - 5211

Pahlavani, P., Delavar, M., Frank, A. (2012) 'Using a Modified Invasive Weed Optimization Algorithm for a Personalized Urban Multi-Criteria Path Optimization Problem'. *International Journal of Applied Earth Observation and Geoinformation*. 18. pp. 313 – 328.

Pinto, L. and Pascoal, M. (2010). 'On Algorithms for the Tri-Criteria Shortest Path Problem with Two Bottleneck Objective Functions'. *Computers and Operations Research*. 37 (1). pp. 1774 - 1779.

Poblete, B., Garcia, R., Mendoza, M. and Jaimes. A. (2011) 'Do All Birds Tweet the Same?: Characterizing Twitter Around The World'. *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM '11)*. Glasgow, United Kingdom, October 24-28, 2011. pp. 1025 – 1030. ACM, ISBN: 9781450307178

Pulido, G. and Coello-Coello, C. (2004) 'The Micro Genetic Algorithm 2: Towards Online Adaptation in Evolutionary Multiobjective Optimisation'. *Technical Report*. Col. San Pedro Zacatenco, Mexico.

Quattrone, A. and Vitetta, A. (2011) 'Random and Fuzzy Utility Models for Road Route Choice'. *Transportation Research Part E: Logistics and Transportation Review*. 47. (6). pp. 1126 – 1139.

Radil, G. (2005). 'Evolutionary Computation: An Overview and Recent Trends'. *ÖGAI Journal*. 24 (1). pp. 2 - 7.

Raith, A. and Ehrgott, M. (2009) 'A Comparison of Solution Strategies for Biobjective Shortest Path Problems'. *Computers and Operations Research*. 36 (4). pp. 1299 – 1331.

Rath, B. and Toth B. (2009) 'Erdos-Renyi Random Graphs + Forest Fires = Self-Organized Criticality'. *Electronic Journal of Probability*. 14. pp. 1290 – 1327.

Ray, T., Gokarn, R. and Sha, O. (1995) 'A Global Optimisation Model for Ship Design'. *Computers in Industry*. 26 (2). pp. 175 – 192.

Rechenberg, I. (1965) 'Cybernetic Solution Path of an Experimental Problem'. Royal Aircraft Establishment Library Translation 1122. Royal Aircraft Establishment, Farnborough.

Ribeiro, B. and Towsley, D. (2010) 'Estimating and Sampling Graphs with Multidimensional Random Walks'. *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (IMC '10)*. Melbourne, Australia - November 1-3, 2010. pp. 390 - 403. ACM, ISBN: 9781450304832

Rickter, K. and Duckham, M. (2008) 'Simplest Instructions: Finding Easy-to-Describe Routes for Navigation'. *Proceedings of the 5th International Conference on Geographic Information Science (GIScience '08)*. Park City, UT, USA, September 23-26, 2008. pp. 274 - 289. Lecture Notes in Computer Science 5266, Springer-Verlag, ISBN: 9783540874720

Ripon, K., Kwong, S. and Man, K. (2007) 'A Real Coding Gene Genetic Algorithm (RJGGA) for multi objective optimisation'. *Information Sciences*. 177 (2). pp . 632 - 654.

Rosentein, M. (1988) 'Data Structures for Programmers'. Wiley-Interscience. ISBN: 9780471635208

Russell, S. and Norvig P. (2003) 'Artificial Intelligence: A Modern Approach' 2nd Edition. Prentice Hall. ISBN: 9780130803023

- Saadatseresht, M., Mansourian, A. and Taleai, M. (2009) 'Evacuation Planning using Multiobjective Evolutionary Optimisation Approach'. *European Journal of Operational Research*. 198 (1). pp. 305 - 314.
- Sanders, P. and Schultes, F. (2005) 'Highway Hierarchies Hasten Exact Shortest Path Queries'. Proceedings of the 13th *European Symposium on Algorithms (ESA2005)*. Palma de Mallorca, Spain, October 3-6, 2005. pp. 568 - 579. Lecture Notes in Computer Science 3669, Springer-Verlang, ISBN: 3540291180
- Santos, J. (2006). 'K Shortest Path Algorithms'. *Proceedings of the Ninth DIMACS Implementation Challenge*. Piscataway, New Jersey, US. November 13-16, 2006. pp 41 – 73. American Mathematical Society, ISBN: 9780821843833.
- Schaffer, J. (1985). 'Multiple Objective Optimisation with Vector Evaluated Genetic Algorithms'. *Proceedings of the 1st International Conference on Genetic Algorithms*. Pittsburgh, Pennsylvania, USA, July 1985. pp. 93 - 100. Lawrence Erlbaum Associates, ISBN: 0805804269
- Schaffer, J. and Eshelman, L. (1991) 'On crossover as an Evolutionarily Viable Strategy'. *Proceedings of the 4th International Conference on Genetic Algorithms*. San Diego, California, USA. July 1991. pp. 61 – 68. Morgan-Kaufmann, ISBN 1558602089
- Sauvanet, G. and Néron, E. (2010) 'Search for the Best Compromise Solution on Multiobjective Shortest Path Problem'. *Electronic Notes in Discrete Mathematics*. 36 (1). pp. 615 - 622.
- Schott, J. (1995) 'Fault Tolerant Design using Single and Multi Criteria Genetic Algorithm Optimisation'. Department of Astronautics and Aeronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA. Unpublished MSc Thesis.
- Schwefel, H. (1998) *Advantages (and Disadvantages) of Evolutionary Computation Over Other Approaches*. In: Back, T., Fogel, D. & Michalewicz, Z. (eds.), *Handbook of Evolutionary Algorithms*. Taylor and Francis. ISBN: 9780750308953
- Schulz, F., Wagner, D. and Zaroliagis, C. (2002) 'Using Multi-Level Graphs for Timetable Information'. *Proceedings of the 4th Workshop on Algorithm Engineering and Experiments*. San Francisco, California, USA, January 4-5, 2002. pp. 43-59. Lecture Notes in Computer Science 2409, Springer, ISBN: 3540439773
- Sedgewicke, R. (2003) *Algorithms in Java, Part 5: Graph Algorithms*. 3rd Edition. Addison-Welsey Professional. ISBN: 9780201361216

Serafini, P. (1994) 'Simulated Annealing for Multiobjective Optimization Problems'. *Proceedings of the 10th International Conference on Multiple Criteria Decision Making: Expand and Enrich the Domains of Thinking and Application*. Taipei, Taiwan July 19-24 1994. pp. 87 - 96. Springer-Verlang, ISBN: 9783540942979

Sergio D. and Guillermo Barrenechea. (2002) 'Constrained Random Walks on Random Graphs: Routing Algorithms for Large Scale Wireless Sensor Networks'. *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA '02)*. Atlanta, Georgia, USA. September 28, 2002. pp. 12 – 21. ACM, ISBN: 1581135890

Shaw, K.; Nortcliffe, A.; Thompson, M.; Love, J.; Fleming, P. and Fonseca, C. (1999) 'Assessing the Performance of Multi Objective Genetic Algorithms for Optimisation of a Batch Process Scheduling Problem'. *Proceedings of the Congress on Evolutionary Computation (CEC1999)*. Washington DC, USA, July 6-9 1999. pp. 37 – 46. IEEE, ISBN: 0780355369

Shi, X., Bonner, M., Adamic, L. and Gilbert. A. (2008) 'The Very Small World of the Well-Connected'. *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia (Hypertext '08)*. Pittsburgh, Pennsylvania, USA, June 19-21, 2008. pp. 61 - 70. ACM, 9781595939852

Shih, L. and Lin, Y. (2003) 'Multi Criteria Optimisation for Infectious Medical Waste Collection System Planning'. *Practical Periodical of Hazardous, Toxic, and Radioactive Waste Management*. 7 (2). pp. 78 - 85.

Shrinvas, N. and Deb, K. (1995) 'Multi Objective Function Optimisation using Non Dominated Sorting Genetic Algorithms'. *Evolutionary Computation*. 2 (3). pp. 221 - 248.

Silva, R. Craveirinha, J and Climaco J (2009) 'Hierarchical Multiobjective Routing in MPLS Networks with Two Service Classes – A Meta-Heuristic Solution'. *Journal of Telecommunications and Information Technology*. 1 (1). pp. 20 - 35.

Singh, H.K., Ray, T. and Smith, W. (2010) 'Surrogate Assisted Simulated Annealing (SASA) for Constrained Multi-Objective Optimization', *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2010)*. Barcelona, Spain, June 18-23 2010, pp. 1 – 8. IEEE, ISBN: 9781424469093

Skriver, A. (2000) 'A Classification of Bicriterion Shortest Path Problems (BSP)'. *Asia-Pacific Journal of Operational Research*. 17 (1). pp. 199 - 212.

Skriver, A. and Andersen, K. (2000). 'A Label Correcting Approach for Solving Bicriterion Shortest-Path Problems'. *Computers and Operations Research*. 27 (6). pp. 507 – 524.

Smith, K. (2006) *Study of Simulated Annealing Techniques for Multi-Objective Optimisation*. Unpublished PhD Thesis, University of Exeter.

Smith, K., Everson, R. Fieldsend, J. Murphy, C. and Misra, R. (2008). 'Dominance-Based Multiobjective Simulated Annealing'. *IEEE Transactions on Evolutionary Computation*. 12 (3). pp. 323 - 342.

Social Research Associates (2005) '*How people choose*', Stage 2B. MR07 Stage One project report to the transport direct programme. Department for Transport (DFT), London, UK.

Spears, W. and DeJong D. (1990) '*An Analysis of Multi-Point Crossover*'. Proceedings of the First Workshop on Foundations of Genetic Algorithms. Bloomington Campus, Indiana, USA, July 15-18 1990. pp. 301 - 315. Morgan-Kaufmann, ISBN: 155860734X

Stewart, B. and White, C. (1991) 'Multiobjective A*'. *Journal of the ACM*. 38 (4). pp. 775 – 814.

Steuer, R. (1986) *Multiple Objective Optimisation: Theory, Computation and Applications*. John Wiley and Sons. ISBN: 9780471888468

Stott, J., Rodgers P., Martinez-Ovando J.C. and Walker. S. (2011) 'Automatic Metro Map Layout Using Multicriteria Optimisation'. *IEEE Transactions on Visualization and Computer Graphics*. 16 (1). pp. 101 - 114.

Suman, B. (2004) 'Study of Simulated Annealing Based Algorithm for Multi Objective Optimisation of a Constrained Problem'. *Computers and Chemical Engineering*. 28 (9). pp. 1849 - 1871.

Suman, B. (2005) 'Study of Self-Stopping PDMOSA and Performance Measure in Multiobjective Optimisation'. *Computers & Chemical Engineering*. 29 (5). pp. 1131 – 1147.

Suman, B. and Kumar, P. (2006) 'A Survey of Simulated Annealing as a Tool for Single and Multiobjective Optimisation'. *Journal of the Operational Research Society*. 57 (10). pp. 1143 – 1160.

Suppaitnarm, A., Seffen, K.A, Parks, G.T. and Clarkson, P.J. (2000) 'A Simulated Annealing Algorithm for Multiobjective Optimisation'. *Engineering Optimisation*. 33. p.59 – 85.

Syswerda, G. (1989) 'Uniform Crossover in Genetic Algorithms'. *Proceedings of the 3rd International Conference on Genetic Algorithms*. George Mason University, Fairfax, Virginia, USA, June 1989. pp 2 - 9. Morgan-Kaufmann, ISBN: 1558600663

Tamaki, H., Mori, M. and Araki, M. (1995) 'Generation of a Set of Pareto-Optimal Solutions by Genetic Algorithms'. *Transactions of the Society of Instrumentation and Control*. 38 (8). pp. 1185 -1192.

Tamaki, H. Kita, H. and Kobayashi, S. (1996) 'Multi Objective Optimisation by Genetic Algorithms: a Review'. *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96)*. May 20-22, 1996, Nayoya University, Japan. pp. 517 - 522. IEEE, ISBN: 0780329023

Taylor, G., Car, A. and Mehner, H. (1999) '*Experimenting with Hierarchical Wayfinding*'. Technical Report, Department of Geomatics, University of Newcastle upon Tyne, United Kingdom.

Terano, T., Takahashi, M. and Hanzawa, K. (2006) 'Multiple Solutions for Plant Design Analyses through a Genetic Algorithm with Tabu Lists'. *Proceedings of the 2006 IEEE International Conference on Systems, Man and Cybernetics (SMC '06)*. Taipai, Taiwan, October 8-11 2006. pp. 3126 - 3131. IEEE, ISBN: 1424400996.

Thorup, M. and Zwick, U. (2005) '*Approximate distance oracles*'. *Journal Of The ACM*. 51 (1). pp. 1 - 24.

Tobita, T. and Kasahara. H. (2002) 'A Standard Task Graph Set for Fair Evaluation of Multiprocessor Scheduling Algorithms'. *Journal of Scheduling*. 5 (5). pp. 379 - 394.

Todd, D. (1997) '*Multiple Criteria Genetic Algorithms in Engineering Design and Operation*'. Unpublished PhD Thesis. University of Newcastle, Newcastle upon Tyne, United Kingdom.

Tretyakov, K., Armas-Cervantes, A., García-Bañuelos, L., Vilo, J. and Dumas, M. (2011) 'Fast Fully Dynamic Landmark-Based Estimation of Shortest Path Distances in Very Large Graphs'. *Proceedings of the 20th ACM International Conference on Information and Knowledge Management (CIKM '11)*. Glasgow, United Kingdom, October 24 - 28, 2011. pp. 1785 - 1794. ACM, ISBN: 9781450307178

Tung C. and Chew K. (1988) 'A Bicriterion Pareto-Optimal Path Algorithm'. *Asia-Pacific Journal of Operations Research*. 5 (1). pp. 166-172.

Tuytens, DA, Teghem J., and El-Sherbeny N. (2003) '*A Particular Multiobjective Vehicle Routing Problem Solved by Simulated Annealing*'. in X. Gandibleux, M. Sevaux, K. Sørensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimisation. Lecture Notes in Economics and Mathematical Systems* 535. pp. 133 – 152. Springer, ISBN: 9783540206378

Twitter (2012) '*Build With Twitter*'. URL: <https://dev.twitter.com/> Accessed: 11/10/2012

Ulbricht, M. (2012) 'Single-Objective vs. Multi-Objective Scheduling Algorithms for Scheduling Jobs in Grid Environment'. *Proceedings of the IEEE 10th International Symposium on Applied Machine Intelligence and Informatics (SAMII)*. Her'any Slovakia, 26-28 Jan. 2012. pp. 411 – 414. IEEE, ISBN: 9781457701955

Ulungu, E.L., Teghem, J. and Ost, C. (1998) 'Interactive Simulated Annealing in a Multiobjective Framework: Application to an Industrial Problem'. *Journal of the Operations Research Society*. 49 (1). pp. 1044 – 1050.

Ulungu, E., Teghaem, J, Fortemps, P.H, and Tuytens, D. (1999). 'MOSA Method: a Tool for Solving Multiobjective Combinatorial Decision Problems'. *Journal of Multi-criteria Decision Analysis*. 8 (4). pp.221 – 236.

Valafar, M., Rejaie, R., Willinger, W., (2009) 'Beyond Friendship Graphs: a Study of User Interactions in Flickr'. *Proceedings of the 2nd ACM workshop on Online social networks (WOSN '09)*. Barcelona, Spain — August 16 - 21, 2009. pp. 25 - 30. ACM, ISBN: 9781605584454

Veldhuzien, D. v. (1999) '*Multi objective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*'. Unpublished Phd Thesis. Graduate School of Engineering, Air Force Institute of Technology, Ohio, USA.

Veldhuizen, D. and Lamont, G. (1999) 'Multi Objective Evolutionary Algorithm Test Suites'. *Proceedings of the Symposium on Applied Computing*. February 28 - March 02, 1999, San Antonio, Texas, USA. pp. 351 - 357. ISBN: 1581130864

Vetter, C. (2010) *Fast and Exact Mobile Navigation with OpenStreetMap Data*, University of the State of Baden-Wuerttemberg. Unpublished Diploma Thesis.

Ware, J.M., Thomas, N. and Jones, C.B. (2002) 'Map Generalisation using Simulated Annealing: Maintaining Feature Alignment'. *Proceedings of the Second International Conference on Geographic Information Science (GIScience '02)*. September 25-28, 2002, Boulder, Colorado, USA. pp. 201 - 203.

Watts, D. and Strogatz, S. (1998) 'Collective dynamics of 'small-world' networks'. *Nature* 393. pp. 440 – 442.

Weiss, M. A. (1994) '*Data Structures and Algorithm Analysis*'. The Benjamin/Cummins Publishing Company. ISBN: 9780321370136

Wen, U., Wang, W. and Yang, C. (2007) 'Traffic Engineering and Congestion Control for Open Shortest Path First Networks'. *Omega*. 35 (6). pp. 671 - 682.

While, L., Bradstreet, L., Barone, L. and Hingston, P. (2005) 'Heuristics for Optimizing the Calculation of Hypervolume for Multi-Objective Optimisation Problems'. *Proceedings Of The 2005 IEEE Congress on Evolutionary Computation*. September 2-4 2005, Edinburgh, United Kingdom. pp. 2225 -2232. IEEE, ISBN: 0780393635.

While, L., Hingston, P., Barone, L. and Huband, S. (2006) 'A Faster Algorithm for Calculating Hypervolume'. *IEEE Transactions on Evolutionary Computation*. 10 (1). pp. 29 - 38.

Winkler, P. (1985) '*Random Orders*'. *Order*. 1 (1). pp. 317 – 331.

Xu, J. and Chen, H. (2004) 'Fighting Organised Crimes: Using Shortest Path Algorithms to Identify Associations in Criminal Networks'. *Decision Support Systems*. 38 (3). pp. 473 - 487.

Yen, J. (1971) 'Finding the K shortest loopless paths in a network'. *Management Science*. 17 (11). pp. 712 – 716.

Yen, J. (1972) 'Another Algorithm for Finding the K Shortest-Loop Less Network Paths'. *Proceedings of the 41st Meetings of the Operations Society of America*. 20 (1).

Zakzouk, A., Zaher, H. and El-Deen, R. (2010) 'An Ant Colony Optimisation Approach for Solving Shortest Path Problem with Fuzzy Constraints'. *Proceedings of the 7th International Conference on Informatics and Systems (INFOS 2010)*. Cairo, Egypt. March 28-30 2010. pp 1 – 8. IEEE, ISBN: 9781424458288

Zengan, G. and Mao, Y. (2007) 'A Framework for Data Mining-Based Anti-Money Laundering Research'. *Journal of Money Laundering Control*. 10 (2). pp. 170 – 179.

Zhan, B. (1997) 'Three Fastest Shortest Path Algorithms on Real Road Networks: Data Structures and Procedures'. *Journal of Geographic Information and Decision Analysis*. 1 (1). pp. 69 - 82.

Zhan, F and Noon C.E. (2000) 'A Comparison Between Label Setting and Label Correcting Algorithms for Computing One-to-One shortest Paths'. *Journal of Geographic Information and Decision Analysis*. 4 (2). pp. 1-11.

Zhang, Y., Gong, D. and Zhang, J. (Online from:2012) 'Robot Path Planning in Uncertain Environment using Multi-Objective Particle Swarm Optimization'. *Neurocomputing*. ISSN 0925-2312. *In Press*.

Zhiping, Q. and Yuxing, Z. (2010) '*Parametric Optimization Design of Aircraft Based on Hybrid Parallel Multi-objective Tabu Search Algorithm*'. *Chinese Journal of Aeronautics*. 23 (4) pp. 430-437

Zhou. A, Qu. B, Li H., Zhao. S, Suganthan. P. and Zhang. Q. (2011) 'Multiobjective Evolutionary Algorithms: A Survey of the State of The Art'. *Swarm and Evolutionary Computation*. 1 (1). pp. 32 - 49.

Zidi, I., Mesghouni, K., Zidi, K. and Ghedira, K. (2011) 'A New Approach Based on the Multi-Objective Simulated Annealing to Solving the Dynamic Dial a Ride Problem'. *Proceedings of the 4th International Conference on Logistics (LOGISTIQUA 2011)*. Hammamet, Tunisia, May 31 2011-June 3 2011 pp. 157 - 163. IEEE, ISBN: 9781457703249

Zijpp, N. and Catalano, S. (2005) 'Path Enumeration by Finding the Constrained K-shortest Paths'. *Transportation Research Part B: Methodological*. 39 (6). pp. 545 – 563.

Zionts S. and Wallenius, J. (1976) 'An Interactive Programming Method for Solving the Multiple Criteria Problem'. *Management Science* 22 (6). pp. 652 - 663.

Zitzler, E. and Thiele. L. (1998) 'Multiobjective Optimisation Using Evolutionary Algorithms - A Comparative Case Study'. *Proceedings of the 5th Annual Conference on Parallel Problem Solving from Nature (PPSN V)*. Amsterdam, The Netherlands, September 27-30 1998, pp. 292–301. Springer-Verlang, ISBN: 3540650784

Zitzler, E. (1999) '*Evolutionary Algorithms for Multi objective Optimisation: Methods and Applications*'. Unpublished Phd Thesis. Computer Engineering and Networks Laboratory. Zurich, Swiss Federal Institute of Technology.

Zitzler, E. and Thiele, L. (1999) 'Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach'. *IEEE Transactions on Evolutionary Computation*. 3 (4) pp.257–271.

Zitzler, E., Laumanns, M. and Thiele, L. (2001) '*SPEA2 : Improving the strength Pareto evolutionary algorithm*'. Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland.

Zitzler, E., Laumanns, M., Thiele, M., Fonseca C. and Grunert da Fonseca. V. (2002) 'Why Quality Assessment Of Multiobjective Optimisers Is Difficult'. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*. New York, New York, USA, July 9-13 2002. pp. 666 – 674. Morgan-Kaufmann, ISBN: 1558608788

Zitzler, E., Laumanns, M and Bleur, S. (2004) '*A Tutorial in Multiobjective Optimisation*'. in *Metaheuristics for Multiobjective Optimisation* (Eds: Gandibleux, X., Saveaux, M., Sorenson, K. and T'Kindt V.). Springer. pp. 3 - 39. ISBN: 9783540206378.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. and Fonseca, V. (2003) 'Performance Assessment of Multiobjective Optimisers: An Analysis and Review'. *IEEE Transactions on Evolutionary Computation* 7 (2). pp. 117 – 131.

Zitzler, E., Brockhoff. D. and Thiele L. (2007) 'The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators via Weighted Integration'. *Proceedings of the Conference on Evolutionary Multi-Criterion Optimisation (EMO 2007)*. Matsushima, Japan, March 5-8, 2007. pp. 862 – 876. Lecture Notes in Computer Science 4403, Springer, ISBN 3540709274

“There ain’t no such thing as a free lunch.”

R. A. Heinlein, *The Moon Is a Harsh Mistress*

Appendix A: Example Random Graphs

c Test Erdos-Reyni Random Graph

a 1 2

a 1 3

a 2 5

a 3 4

a 4 1

a 4 5

a 4 2

a 5 3

Table A.1 Simple Example of Erdos-Reyni Random Graph (n=5, m=8)

c Test Gilbert Graph With P = 0.1

a 1 4

Table A.2 Example of Gilbert Graph with p=0.1

c Test Gilbert Graph With P = 0.6

a 1 2

a 1 3

a 2 3

a 2 5

a 3 4

a 4 5

Table A.3 Example of Gilbert Graph with p=0.6

c Test Gilbert Graph With P = 0.8

a 1 2

a 1 3

a 1 4

a 1 5

a 2 3

a 2 4

a 2 5

a 3 4

a 3 5

a 4 5

Table A.4 Example of Gilbert Graph with p=0.8

c Watts And Strogatz Graph {10,0.0,4}
a 1 2
a 1 3
a 2 3
a 2 4
a 3 4
a 3 5
a 4 5
a 4 6
a 5 6
a 5 7
a 6 7
a 6 8
a 7 8
a 7 9
a 8 9
a 8 10
a 9 10
a 9 1
a 10 1
a 10 2

Table A.5 Watts and Strogatz Example Graph (p=0.0, k = 4, n=10)

c Watts And Strogatz Graph {10,0.5,4}
a 1 10
a 1 3
a 2 3
a 2 4
a 3 4
a 3 5
a 4 8
a 4 7
a 5 9
a 5 4
a 6 9
a 6 5
a 7 8
a 7 9
a 8 9
a 8 10
a 9 1
a 9 10
a 10 7
a 10 2

Table A.6 Watts and Strogatz example graph (p=0.5, k = 4, n=10)

c Watts And Strogatz Graph {10,0.8,4}	
a	1 5
a	1 6
a	2 7
a	2 1
a	3 4
a	3 10
a	4 7
a	4 5
a	5 2
a	5 9
a	6 2
a	6 10
a	7 5
a	7 6
a	8 10
a	8 4
a	9 8
a	9 2
a	10 7
a	10 4

Table A.7 Watts and Strogatz Example Graph (p=0.8, k = 4, n=10)

c Barabasi-Ablert Model Initial Network {3,10,2}	
a	1 2
a	1 3
a	1 4
a	2 3
a	2 4
a	3 4

Table A.8 Barabasi and Albert Initial Random Graph

c	Barbasi-Albert Graph {3,10,2}
a	1 2
a	1 3
a	1 4
a	2 3
a	2 4
a	3 4
a	5 3
a	5 4
a	6 5
a	6 1
a	7 3
a	7 4
a	8 4
a	8 1
a	9 4
a	9 5
a	10 6
a	10 7

Table A.9 Barabasi and Albert Complete Random Graph

Appendix B: Result Tables

	Runtime In Seconds Of Heuristic Approaches				
	GA (RW)	GA (KG)	TS	SA	PAES
100 X 200	14.22	1.8	12.56	1.7	1.4
200 X 400	14.35	2.1	15.89	2.3	2.2
500 X 1000	14.59	2.3	18.89	3.4	3.1
750 X 1500	14.66	2.8	21.34	5.2	5.6
1000 X 2000	14.67	3.1	23.89	7.9	7.8
2000 X 4000	14.80	3.3	25.21	17.4	17.8
3000 X 6000	15.12	3.5	28.78	48.9	46.2
5000 X 10000	15.15	3.9	32.13	84.45	66.3
6000 X 12000	16.35	4.2	35.78	119	128
8000 X 16000	19.1	4.6	38.32	261	254
10000 X 20000	20.25	4.9	43.21	331	329
12000 X 24000	21.6	5.1	46.51	342	348

Table B.1 Runtime of Heuristics on Random Graphs

	Runtime In Seconds Of Heuristic Approaches				
	GA (RW)	GA(KG)	TS	SA	PAES
Cardiff 500	21.1	1.2	27.3	92.321	94.543
Cardiff 750	55.91	3.7	73.35	283.04	286.43
Cardiff 1000	139.63	6.9	181.34	854.23	867.45
Cardiff 2000	1407	26.1	1837.12	-	-
London 500	31.6	4.7	51.4	176.34	178.87
London 1000	109.22	7.9	206.124	901.34	896.45
London 2000	1845	14.1	2388	-	-
NE 1000	10.9	12.7	14.5	56.34	57.12
NE 2000	45.54	36.1	56.54	274.28	281.45

Table B.2 Runtime of Heuristics on Real World Graphs

Graph	Tests	$D(PfAPPROX; PfTRUE) \leq 0$	$D(PfAPPROX; PfTRUE) = 1$	$D(PfAPPROX; PfTRUE) \geq 2$
100 X 150	100	100	0	0
100 X 200	100	100	0	0
100 X 300	100	99	1	0
200 X 300	100	100	0	0
200 X 400	100	100	0	0
500 X 1000	100	98	0	2
500 X 1500	100	99	0	1
750 X 1500	100	96	1	3
750 X 2000	100	95	2	3
1000 X 1500	100	91	4	5
1000 X 2000	100	90	3	7
1000 X 3000	100	94	1	5
2000 X 3000	100	88	2	10
2000 X 4000	100	87	5	8
2000 X 5000	100	87	4	9
3000 X 5000	100	83	6	11
3000 X 6000	100	81	4	15
5000 X 6000	100	80	9	11
5000 X 7000	100	79	7	14
5000 X 8000	100	74	5	21
5000 X 10000	100	76	12	12
5000 X 15000	100	77	11	12
8000 X 9000	100	70	15	15
8000 X 12000	100	67	9	24
10000 X 15000	100	63	16	21
10000 X 20000	100	61	13	26

Table B.3 Summary of Results using GA (Random Walk Based)

Graph	Tests	$D(PfAPPROX; PfTRUE) \leq 0$	$D(PfAPPROX; PfTRUE) = 1$	$D(PfAPPROX; PfTRUE) \geq 2$
100 X 150	100	100	0	0
100 X 200	100	100	0	0
100 X 300	100	100	0	0
200 X 300	100	100	0	0
200 X 400	100	100	0	0
500 X 1000	100	100	0	0
500 X 1500	100	100	0	0
750 X 1500	100	99	1	0
750 X 2000	100	98	1	1
1000 X 1500	100	97	1	2
1000 X 2000	100	96	2	2
1000 X 3000	100	93	2	5
2000 X 3000	100	92	5	3
2000 X 4000	100	89	5	6
2000 X 5000	100	83	7	10
3000 X 5000	100	81	9	11
3000 X 6000	100	79	11	10
5000 X 6000	100	73	10	27
5000 X 7000	100	69	10	22
5000 X 8000	100	67	10	13
5000 X 10000	100	63	8	29
5000 X 15000	100	62	11	27
8000 X 9000	100	56	20	24
8000 X 12000	100	53	19	28
10000 X 15000	100	54	19	27
10000 X 20000	100	52	18	30

Table B.4 Summary of Results using GA (K Geodesic Based)

Graph	Tests	$D(PfAPPROX; PfTRUE) = 0$	$D(PfAPPROX; PfTRUE) = 1$	$D(PfAPPROX; PfTRUE) \geq 2$
100 X 150	100	95	4	1
100 X 200	100	96	2	2
100 X 300	100	88	6	6
200 X 300	100	82	10	8
200 X 400	100	85	11	4
500 X 1000	100	76	14	10
500 X 1500	100	78	13	19
750 X 1500	100	74	10	16
750 X 2000	100	72	9	19
1000 X 1500	100	69	15	16
1000 X 2000	100	77	7	16
1000 X 3000	100	72	12	14
2000 X 3000	100	75	15	10
2000 X 4000	100	70	15	15
2000 X 5000	100	67	14	19
3000 X 5000	100	65	13	22
3000 X 6000	100	72	17	11
5000 X 6000	100	65	18	17
5000 X 7000	100	81	10	9
5000 X 8000	100	61	29	10
5000 X 10000	100	52	11	33
5000 X 15000	100	54	24	22
8000 X 9000	100	51	22	27
8000 X 12000	100	43	34	23
10000 X 15000	100	47	24	29
10000 X 20000	100	47	27	26

Table B.5 Summary of Quality Tests using PAES

Graph	Tests	$D(PfAPPROX; PfTRUE) \leq 0$	$D(PfAPPROX; PfTRUE) = 1$	$D(PfAPPROX; PfTRUE) \geq 2$
100 X 150	100	85	6	9
100 X 200	100	86	7	7
100 X 300	100	84	5	11
200 X 300	100	86	8	6
200 X 400	100	82	11	7
500 X 1000	100	76	14	10
500 X 1500	100	78	13	9
750 X 1500	100	73	10	17
750 X 2000	100	72	9	19
1000 X 1500	100	70	14	16
1000 X 2000	100	71	14	15
1000 X 3000	100	70	13	17
2000 X 3000	100	74	16	10
2000 X 4000	100	69	15	16
2000 X 5000	100	67	18	15
3000 X 5000	100	63	15	22
3000 X 6000	100	71	18	11
5000 X 6000	100	66	17	19
5000 X 7000	100	58	15	27
5000 X 8000	100	61	29	10
5000 X 10000	100	52	13	35
5000 X 15000	100	54	24	22
8000 X 9000	100	51	21	28
8000 X 12000	100	47	31	22
10000 X 15000	100	43	26	31
10000 X 20000	100	47	27	26

Table B.6 Summary of Quality Tests Using Tabu Search

Graph	Tests	$D(Pf_{APPROX}; Pf_{TRUE}) \leq 0$	$D(Pf_{APPROX}; Pf_{TRUE}) = 1$	$D(Pf_{APPROX}; Pf_{TRUE}) \geq 2$
100 X 150	100	91	5	4
100 X 200	100	88	6	6
100 X 300	100	87	7	6
200 X 300	100	84	6	10
200 X 400	100	82	11	7
500 X 1000	100	76	14	10
500 X 1500	100	78	13	19
750 X 1500	100	73	10	17
750 X 2000	100	72	9	19
1000 X 1500	100	69	15	16
1000 X 2000	100	71	14	15
1000 X 3000	100	72	12	14
2000 X 3000	100	75	15	10
2000 X 4000	100	69	15	16
2000 X 5000	100	66	18	16
3000 X 5000	100	65	13	22
3000 X 6000	100	71	15	14
5000 X 6000	100	65	18	17
5000 X 7000	100	65	19	16
5000 X 8000	100	61	29	10
5000 X 10000	100	52	11	33
5000 X 15000	100	54	24	22
8000 X 9000	100	53	20	27
8000 X 12000	100	45	32	23
10000 X 15000	100	46	24	30
10000 X 20000	100	49	26	25

Table B.7 Summary of Quality Tests Using Simulated Annealing

Graph	GA (K Geodesic)	GA (RW)	TS	SA	PAES
100 X 150	6 (100)	6 (100)	7 (85)	6 (91)	11 (95)
100 X 200	12 (100)	9 (100)	12 (86)	13 (88)	17 (96)
100 X 300	9 (100)	11 (99)	9 (84)	12 (87)	13 (88)
200 X 300	7 (100)	14 (100)	14 (86)	16 (84)	12 (82)
200 X 400	13 (100)	8 (100)	11 (82)	9 (82)	14 (85)
500 X 1000	9 (100)	21 (98)	5 (76)	12 (76)	8 (76)
500 X 1500	5 (100)	3 (99)	8 (78)	14 (78)	11 (78)
750 X 1500	11 (99)	12 (96)	9 (73)	11 (73)	8 (74)
750 X 2000	7 (98)	8 (95)	13 (72)	12 (72)	14 (72)
1000 X 1500	8 (97)	11 (91)	12 (70)	12 (69)	15 (69)
1000 X 2000	15 (96)	7 (90)	13 (71)	11 (71)	12 (77)
1000 X 3000	8 (93)	15 (94)	11 (70)	14 (72)	9 (72)
2000 X 3000	11 (92)	10 (88)	8 (74)	12 (75)	9 (75)
2000 X 4000	4 (89)	8 (87)	11 (69)	11 (69)	10 (70)
2000 X 5000	12 (83)	13 (87)	12 (67)	13 (66)	8 (67)
3000 X 5000	11 (81)	12 (83)	14 (63)	12 (65)	12 (65)
3000 X 6000	14 (79)	9 (81)	10 (71)	15 (71)	11 (72)
5000 X 6000	21 (73)	11 (80)	7 (66)	9 (65)	12 (65)
5000 X 7000	16 (69)	10 (79)	8 (58)	12 (65)	16 (81)
5000 X 8000	11 (67)	13 (74)	11 (61)	13 (61)	14 (61)
5000 X 10000	5 (63)	6 (76)	9 (52)	11 (52)	9 (52)
5000 X 15000	8 (62)	8 (77)	13 (54)	7 (54)	9 (54)
8000 X 9000	7 (56)	11 (70)	12 (51)	8 (53)	13 (51)
8000 X 12000	12 (53)	9 (67)	12 (47)	7 (45)	14 (43)
10000 X 15000	10 (54)	15 (63)	4 (43)	11 (46)	9 (47)
10000 X 20000	14 (52)	8 (61)	6 (47)	9 (49)	10 (47)

Table B.8 Presence of Local Optimal Paths for Each Algorithm

Graph	GA (K Geodesic)	GA (RW)	TS	SA	PAES
100 X 150	3.25	3.45	4.13	3.43	4.78
100 X 200	3.2	3.36	3.17	5.3	8.24
100 X 300	2.2	4.12	5.53	1.9	4.7
200 X 300	6.21	4.32	7.86	3.69	5.12
200 X 400	4.73	5	5.26	4.6	3.3
500 X 1000	3.27	8.12	4.31	5.21	2.78
500 X 1500	5.28	4.47	5.32	4.78	6.01
750 X 1500	3.21	3.89	2.34	2.21	3.54
750 X 2000	2.26	4	2.38	3.56	5.21
1000 X 1500	6.67	4.19	5.03	3.56	3.6
1000 X 2000	1.98	2.38	2.21	2.28	3.2
1000 X 3000	2.6	3.34	2.23	4.31	2.27
2000 X 3000	2.8	2.21	3.38	4.17	3.41
2000 X 4000	4.14	3.45	6.23	5.42	4.33
2000 X 5000	3.89	2.29	3.21	4.32	4.21
3000 X 5000	2.34	3	3.29	2.12	2.23
3000 X 6000	1.58	5.6	2.21	4.56	3.2
5000 X 6000	3.21	2.87	3.31	2.56	3.45
5000 X 7000	3.44	3.56	4.21	7.21	4.11
5000 X 8000	2.3	3.31	1.98	3.43	5.1
5000 X 10000	6.34	3.21	2.32	7.01	2.26
5000 X 15000	3.23	3.54	4.03	2.92	3.66
8000 X 9000	5.21	3.21	3.45	8.01	3.77
8000 X 12000	3.52	5.26	2.26	3.41	3.44
10000 X 15000	4.77	3.41	4.32	2.56	3.21
10000 X 20000	2.29	4.69	5.12	2.21	4.98

Table B.9 Average Distance of Locally Optimal Solutions from PfTRUE

Appendix C: UML Diagrams

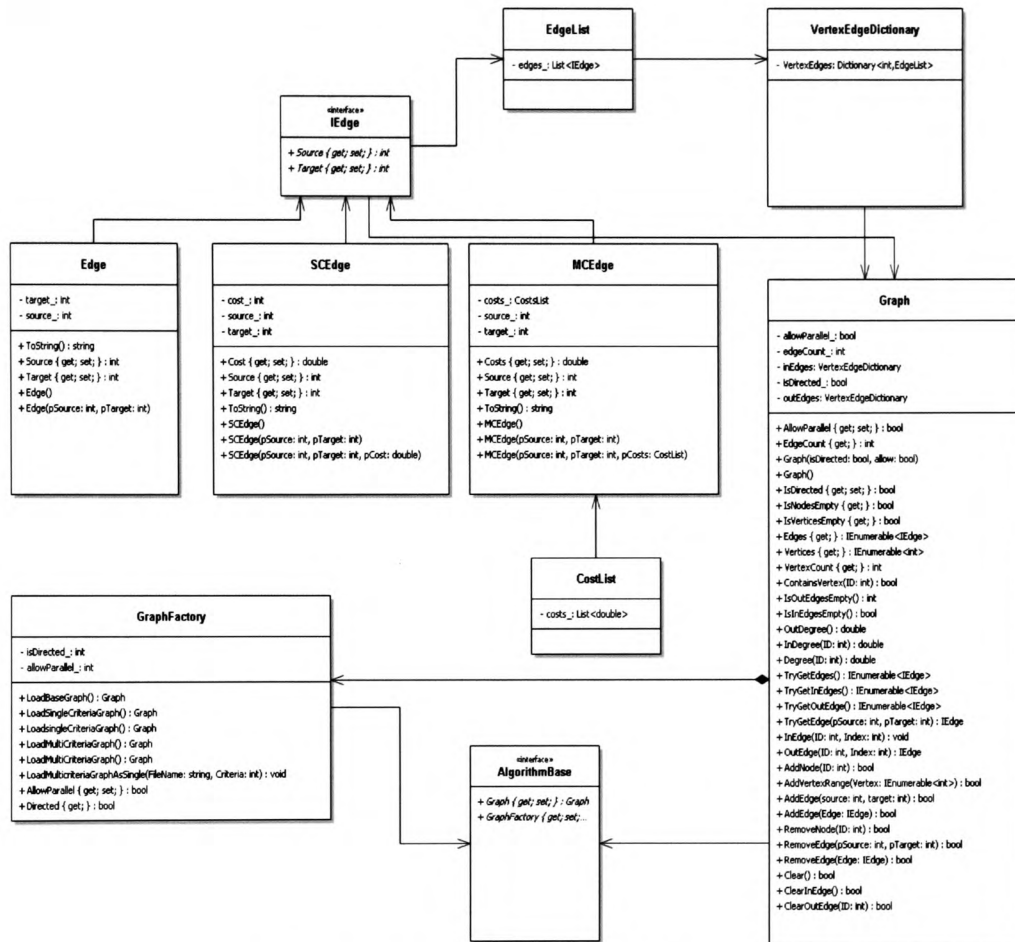


Figure C.1 UML Diagram For Graph Structure

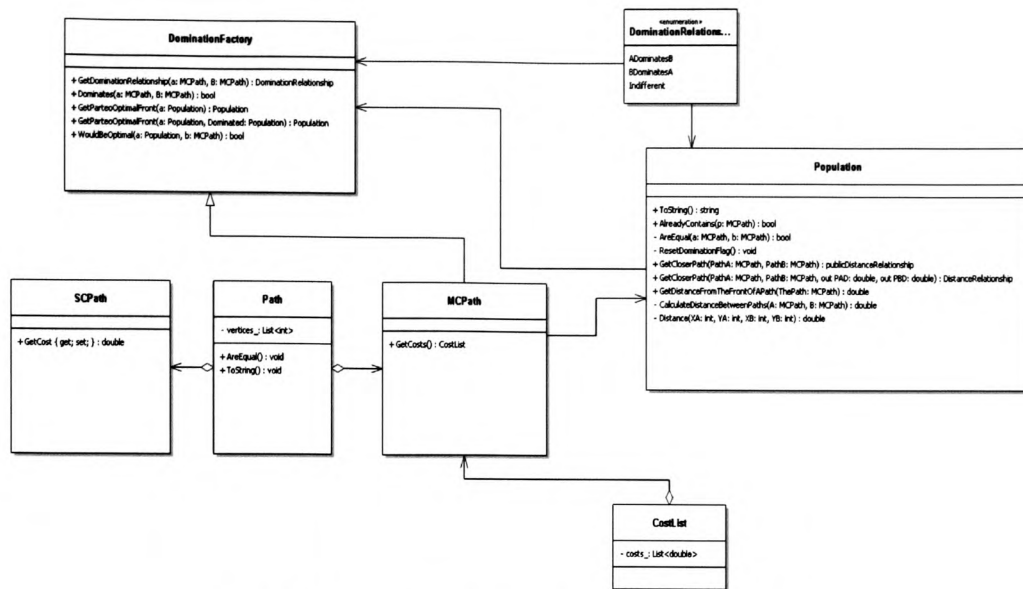


Figure C.2 UML Diagram for Domination System

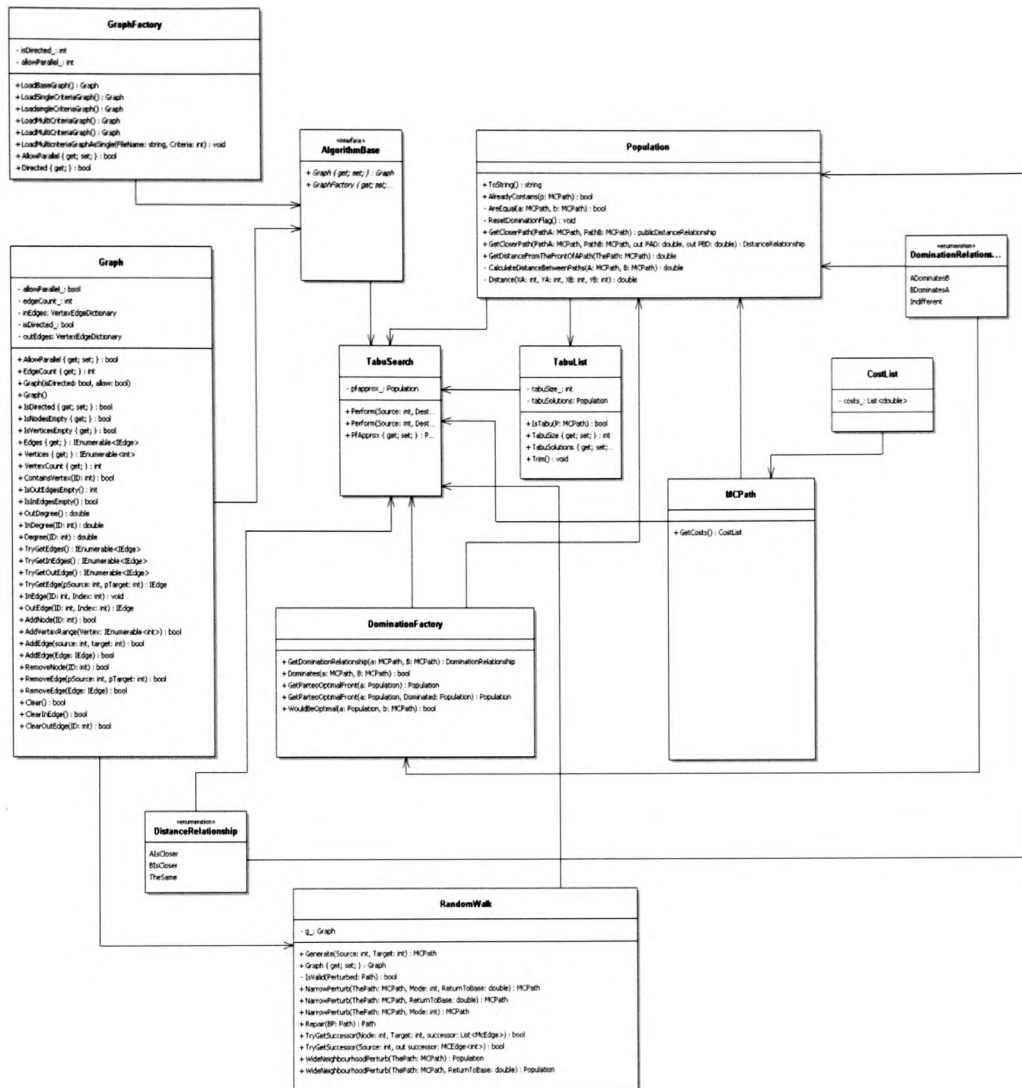


Figure C.4 UML Diagram for Example Algorithm (Tabu Search)