

**ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE**  
**ENGINEERING AND TECHNOLOGY**

**ENHANCED HYBRID BIG BANG-BIG CRUNCH OPTIMIZATION  
ALGORITHMS AND APPLICATIONS ON SINGLE AND MULTI-OBJECTIVE  
AIRPORT GATE ASSIGNMENT PROBLEM**

**Ph.D. THESIS**

**Hakkı Murat GENÇ**

**Department of Control and Automation Engineering**

**Control and Automation Engineering Programme**

**JUNE 2012**



**ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL OF SCIENCE**  
**ENGINEERING AND TECHNOLOGY**

**ENHANCED HYBRID BIG BANG-BIG CRUNCH OPTIMIZATION  
ALGORITHMS AND APPLICATIONS ON SINGLE AND MULTI-OBJECTIVE  
AIRPORT GATE ASSIGNMENT PROBLEM**

**Ph.D. THESIS**

**Hakkı Murat GENÇ**  
**504062101**

**Department of Control and Automation Engineering**

**Control and Automation Engineering Programme**

**Thesis Advisor: Prof. Dr. İbrahim EKSİN**  
**Assist. Prof. Dr. Osman Kaan Erol**

**JUNE 2012**



**İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ**

**GELİŞTİRİLMİŞ MELEZ BÜYÜK PATLAMA-BÜYÜK ÇÖKÜŞ  
OPTİMİZASYON ALGORİTMALARI VE TEK VE ÇOK AMAÇLI  
HAVAALANI KAPI ATAMA PROBLEMİ UYGULAMALARI**

**DOKTORA TEZİ**

**Hakkı Murat GENÇ  
504062101**

**Kontrol ve Otomasyon Mühendisliği Anabilim Dalı**

**Kontrol ve Otomasyon Mühendisliği Programı**

**Tez Danışmanı: Prof. Dr. İbrahim EKSİN  
Yrd. Doç. Dr. Osman Kaan Erol**

**HAZİRAN 2012**









*To my spouse and child,*



## FOREWORD

I would like to sincerely acknowledge my advisor Prof.Dr.İbrahim Eksin for his friendly and wise inspiration. He was the *local search* effect in this study. My co-advisor, Assist.Prof.Dr.Osman Kaan Erol was equally friendly and contributive with his brilliant ideas. He was certainly the *global search* effect in the thesis scope.

I also wish to thank Prof.Dr.Müjde Güzelkaya, Assist.Prof.Dr.Taner Arsan and Assist.Prof.Dr.Ahmet Onat for the inspiring discussions.

Lastly, I have a heart full of thanks for my beloved wife, Nevra, for tolerating the times I stole from us for my education and being the biggest motivation in my life.

June 2012

Hakkı Murat Genç



## TABLE OF CONTENTS

|   | <u>Page</u>  |
|---|--------------|
| <b>FOREWORD</b> .....   | <b>ix</b>    |
| <b>TABLE OF CONTENTS</b> .....  | <b>xi</b>    |
| <b>ABBREVIATIONS</b> .....  | <b>xv</b>    |
| <b>LIST OF TABLES</b> .....   | <b>xvii</b>  |
| <b>LIST OF FIGURES</b> .....  | <b>xix</b>   |
| <b>SUMMARY</b> .....  | <b>xxi</b>   |
| <b>ÖZET</b> .....   | <b>xxiii</b> |
| <b>1. INTRODUCTION</b> .....  | <b>27</b>    |
| 1.1 What is an Evolutionary Algorithm? .....  | 27           |
| 1.2 A Brief History on Evolutionary Algorithms .....                                      | 28           |
| 1.3 Airport Gate Assignment Problem .....   | 29           |
| 1.4 Purpose of Thesis .....   | 29           |
| 1.5 A Brief Summary of Chapters .....   | 29           |
| <b>2. BIG BANG-BIG CRUNCH ALGORITHM WITHIN EVOLUTIONARY<br/>COMPUTATION METHODS</b> ..... | <b>31</b>    |
| 2.1 Components of the Evolutionary Algorithms .....                                       | 31           |
| 2.1.1 Representation .....  | 31           |
| 2.1.2 Objective function .....  | 31           |
| 2.1.3 Population .....  | 32           |
| 2.1.4 Parent selection .....  | 32           |
| 2.1.5 Variation operators .....   | 32           |
| 2.1.6 Survivor selection .....  | 33           |
| 2.2 Algorithms Reported in the Literature .....   | 33           |
| 2.2.1 Genetic algorithms .....  | 33           |
| 2.2.1.1 Representation in GA .....  | 33           |
| 2.2.1.2 Population in GA .....  | 34           |
| 2.2.1.3 Parent selection in GA .....  | 34           |
| 2.2.1.4 Variation operators in GA .....   | 35           |
| 2.2.1.5 Recombination operators .....   | 35           |
| 2.2.1.6 Mutation operators .....  | 38           |
| 2.2.1.7 Survivor selection in GA .....  | 39           |
| 2.2.2 Evolution strategies .....  | 40           |
| 2.2.2.1 Representation in ES .....  | 40           |
| 2.2.2.2 Population in ES .....  | 41           |
| 2.2.2.3 Parent selection in ES .....  | 41           |
| 2.2.2.4 Variation operators in ES .....   | 41           |
| 2.2.2.5 Recombination operators .....   | 41           |
| 2.2.2.6 Survivor selection in ES .....  | 46           |
| 2.2.2.7 Self adaptation in ES .....   | 46           |
| 2.2.3 Big Bang – Big Crunch optimization algorithm .....                                  | 47           |

|   |            |
|---|------------|
| 2.2.4 Relations and differences with the previously reported literature .....   | 49         |
| 2.2.4.1 Similar genetic algorithm approaches .....  | 50         |
| 2.2.4.2 Similar evolutionary strategies approaches .....  | 50         |
| <b>3. BIG BANG – BIG CRUNCH OPTIMIZATION WITH LOCAL DIRECTIONAL MOVES .....</b>   | <b>53</b>  |
| 3.1 Local Search in Evolutionary Computation .....  | 53         |
| 3.2 Inspection of the Effect of Nelder-Mead Crunching .....   | 55         |
| 3.2.1 Big bang-big crunch algorithm with Nelder-Mead crunching .....  | 55         |
| 3.2.2 Simulation results for Nelder Mead crunching.....   | 58         |
| 3.3 Inspection of the Effect of Improvement Vectors.....  | 61         |
| 3.3.1 Big Bang-Big Crunch algorithm with improvement vectors.....   | 61         |
| 3.3.1.1 Vector formation with single step regression .....  | 62         |
| 3.3.1.2 Vector formation with double step regression .....  | 64         |
| 3.3.1.3 Dichotomous search on local direction vector.....   | 65         |
| 3.3.2 Simulation results for improvement vector generation.....   | 66         |
| 3.4 BB – BC with Local Directional Moves (BBBC – LS).....   | 75         |
| 3.4.1 Algorithm formulation .....   | 75         |
| 3.4.2 Simulation results for BBBC-LS .....  | 77         |
| 3.5 Conclusion.....   | 82         |
| <b>4. SINGLE LEAP-BIG BANG BIG CRUNCH OPTIMIZATION APPROACH TO SINGLE OBJECTIVE AIRPORT GATE ASSIGNMENT PROBLEM.....</b>  | <b>85</b>  |
| 4.1 Introduction .....  | 85         |
| 4.2 Problem Formulation .....   | 87         |
| 4.3 Heuristic and Optimization Based Solution Approaches.....   | 89         |
| 4.3.1 Heuristic approaches .....  | 90         |
| 4.3.1.1 A previously reported heuristic: Greedy algorithm for minimizing the number of flights assigned to the apron..... | 90         |
| 4.3.1.2 A new heuristic approach: Ground time duration maximization algorithm (GTMA) .....                                | 91         |
| 4.3.2 Single Leap-Big Bang Big Crunch algorithm (SL- BBBC).....   | 92         |
| 4.4 Simulation Results .....  | 95         |
| 4.4.1 Simulation results with artificially generated dataset .....  | 95         |
| 4.4.2 Simulation results with actual field data .....   | 100        |
| 4.5 Application at Atatürk Airport of İstanbul.....   | 102        |
| 4.6 Conclusion.....   | 107        |
| <b>5. INTRODUCTION TO MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS .....</b>   | <b>109</b> |
| 5.1 Multi-Objective Problem (MOP) Definitions and Basic Concepts .....  | 109        |
| 5.2 Classification of MOP Solution Techniques .....   | 110        |
| 5.3 Basic Concepts on Multi-Objective Evolutionary Algorithms (MOEAs) .....   | 111        |
| 5.4 Pareto Based MOEA Concepts.....   | 112        |
| 5.4.1 Dominance-based ranking or fitness assignment.....  | 113        |
| 5.4.2 Diversity preservation .....  | 114        |
| 5.5 MOEA Population Structure .....   | 117        |
| 5.6 Baseline Algorithms.....  | 118        |
| 5.6.1 Nondominated sorting genetic algorithm-II .....   | 118        |
| 5.6.2 Pareto archived evolution strategy.....   | 120        |
| 5.6.3 Strength pareto evolutionary algorithm-II.....  | 122        |
| 5.7 MOEA Testing.....   | 124        |
| 5.7.1 Basic test suites for multi-objective evolutionary algorithms .....   | 125        |

|   |            |
|---|------------|
| 5.8 Metrics of Performance .....  | 126        |
| <b>6. ENHANCED ORDER BASED SINGLE LEAP-BIG BANG BIG CRUNCH<br/>OPTIMIZATION APPROACH TO MULTI-OBJECTIVE AIRPORT GATE<br/>ASSIGNMENT PROBLEM .....</b> | <b>131</b> |
| 6.1 Introduction.....   | 131        |
| 6.2 Problem Formulation.....  | 132        |
| 6.3 Multi-Objective Gate Assignment Problem Solution Techniques.....  | 134        |
| 6.3.1 Enhanced order based SL-BBBC algorithm (eSL-BBBC):.....   | 137        |
| 6.3.1.1 Creating the initial population.....  | 138        |
| 6.3.1.2 Neighbor generation .....   | 139        |
| 6.3.1.3 Assignment of planes.....   | 139        |
| 6.3.1.4 Acceptance of the neighbors .....   | 140        |
| 6.3.1.5 Archive postprocessing.....   | 142        |
| 6.3.1.6 Stopping criterion .....  | 143        |
| 6.4 Simulation Results.....   | 144        |
| 6.4.1 Problem instance generation .....   | 144        |
| 6.4.1.1 Flight generation.....  | 144        |
| 6.4.1.2 Airport topology and walking distances .....  | 144        |
| 6.4.1.3 Passenger flow model .....  | 146        |
| 6.4.1.4 Preference model .....  | 148        |
| 6.4.2 Experiments on artificial data .....  | 148        |
| 6.4.3 Experiments on actual field data .....  | 151        |
| 6.5 Conclusions.....  | 152        |
| <b>7. CONCLUSIONS AND RECOMMENDATIONS.....</b>  | <b>155</b> |
| <b>REFERENCES .....</b>   | <b>157</b> |
| <b>CURRICULUM VITAE.....</b>  | <b>167</b> |





## ABBREVIATIONS

|                  |   |
|------------------|---|
| <b>BB-BC</b>     | : Big Bang-Big Crunch   |
| <b>BBBC-LS</b>   | : Big Bang-Big Crunch Algorithm with Local Directional Moves        |
| <b>CEC</b>       | : Congress on Evolutionary Computation                              |
| <b>CMA-ES</b>    | : Covariance Matrix Adaptation Evolution Strategy                   |
| <b>CPU</b>       | : Central Processing Unit   |
| <b>DM</b>        | : Decision Making / Maker   |
| <b>DTLZ</b>      | : Deb-Thiele-Laumanns-Zitzler                                       |
| <b>EA</b>        | : Evolutionary Algorithms   |
| <b>ELECTRE</b>   | : Elimination and Choice Translating Algorithm                      |
| <b>ES</b>        | : Evolutionary Strategies   |
| <b>eSL-BBBC</b>  | : Enhanced Order Based Single Leap-Big Bang Big Crunch Algorithm    |
| <b>FE</b>        | : Function Evaluation   |
| <b>FIFO</b>      | : First In First Out  |
| <b>FPS</b>       | : Fitness Proportional Selection                                    |
| <b>GA</b>        | : Genetic Algorithm   |
| <b>GAP</b>       | : Gate Assignment Problem   |
| <b>GTMA</b>      | : Ground Time Duration Maximization Algorithm                       |
| <b>HR</b>        | : Hyper-Volume Ratio  |
| <b>IEEE</b>      | : The Institute of Electrical and Electronics Engineers             |
| <b>IV</b>        | : Improvement Vector  |
| <b>LIP</b>       | : Local Improvement Process   |
| <b>LLS</b>       | : Local Improvement Process Oriented Local Search                   |
| <b>LS</b>        | : Local Search  |
| <b>MA</b>        | : Memetic Algorithm   |
| <b>MAC</b>       | : Multi-parent Arithmetic Crossover                                 |
| <b>MO</b>        | : Multi-Objective   |
| <b>MOGA</b>      | : Multi-Objective Genetic Algorithm                                 |
| <b>MOGAP</b>     | : Multi-Objective Gate Assignment Problem                           |
| <b>MOEA</b>      | : Multi-Objective Evolutionary Algorithm                            |
| <b>MOP</b>       | : Multi-Objective Optimization Problem                              |
| <b>NPGA</b>      | : Niche-Pareto Genetic Algorithm                                    |
| <b>NSGA</b>      | : Nondominated Sorting Genetic Algorithm                            |
| <b>NM</b>        | : Nelder-Mead   |
| <b>NP-hard</b>   | : Non-Deterministic Polynomial-Time Hard                            |
| <b>ONVG</b>      | : Overall Non-Dominated Vector Generation                           |
| <b>PAES</b>      | : Pareto Archived Evolution Strategy                                |
| <b>P</b>         | : Pareto Optimal Set  |
| <b>PF</b>        | : Pareto Front  |
| <b>PMX</b>       | : Partially Mapped Crossover  |
| <b>PROMETHEE</b> | : Preference Ranking Organization Method for Enrichment Evaluations |
| <b>PSA</b>       | : Pareto Simulated Annealing  |

|                |  |
|----------------|--|
| <b>PSO</b>     | : Particle Swarm Optimization              |
| <b>RMS</b>     | : Resource Management System               |
| <b>RS</b>      | : Ranking Selection                        |
| <b>RWS</b>     | : Roulette Wheel Selection                 |
| <b>SOGAP</b>   | : Single Objective Gate Assignment Problem |
| <b>Spc</b>     | : Spacing                                  |
| <b>SPEA</b>    | : Strength Pareto Evolutionary Algorithm   |
| <b>SUS</b>     | : Stochastic Universal Sampling            |
| <b>SL-BBBC</b> | : Single Leap Big Bang-Big Crunch          |
| <b>XLS</b>     | : Crossover-Based Local Search             |
| <b>ZDT</b>     | : Zitzler-Deb-Thiele                       |

## LIST OF TABLES

|   | <u>Page</u> |
|---|-------------|
| <b>Table 1.1</b> : Evolutionary computing metaphor .....  | 27          |
| <b>Table 2.1</b> : Average costs for multimodal test functions .....  | 52          |
| <b>Table 3.1</b> : Ackley test function results.....  | 59          |
| <b>Table 3.2</b> : Rastrigin test function results.....   | 60          |
| <b>Table 3.3</b> : Rosenbrock test function results .....   | 60          |
| <b>Table 3.4</b> : Ackley test function results for the second termination criterion .....  | 61          |
| <b>Table 3.5</b> : Ackley test function.....  | 68          |
| <b>Table 3.6</b> : Ellipsoid test function .....  | 69          |
| <b>Table 3.7</b> : Rastrigin test function.....   | 70          |
| <b>Table 3.8</b> : Rosenbrock test function .....   | 71          |
| <b>Table 3.9</b> : Sphere test function .....   | 72          |
| <b>Table 3.10</b> : Step test function .....  | 73          |
| <b>Table 3.11</b> : Average costs for all functions.....  | 74          |
| <b>Table 3.12</b> : Definition of algorithm parameters .....  | 77          |
| <b>Table 3.13</b> : Summary of the benchmark functions.....   | 79          |
| <b>Table 3.14</b> : Average performance scores .....  | 80          |
| <b>Table 3.15</b> : Order of algorithms (1: Best, 2: Second, 3: Third, 4: Worst) .....  | 81          |
| <b>Table 3.16</b> : Summary of algorithm comparison.....  | 81          |
| <b>Table 3.17</b> : Complexity Analysis .....   | 82          |
| <b>Table 4.1</b> : Mean cost values for synthetic dataset .....   | 98          |
| <b>Table 4.2</b> : Mean cost values for 31 days .....   | 101         |
| <b>Table 4.3</b> : Mean cost values for the data collected from resource management<br>system of Atatürk Airport for month February 2010..... | 106         |
| <b>Table 5.1</b> : Pseudocode for NSGA-II.....  | 120         |
| <b>Table 5.2</b> : Pseudocode for (1+1) PAES .....  | 121         |
| <b>Table 5.3</b> : Pseudocode for archiving test in PAES .....  | 122         |
| <b>Table 5.4</b> : Pseudocode for archiving test in SPEA-II .....   | 124         |
| <b>Table 6.1</b> : Archive postprocessing algorithm.....  | 142         |
| <b>Table 6.2</b> : MOGAP solution.....  | 143         |
| <b>Table 6.3</b> : Parameters of the artificial data sets.....  | 149         |
| <b>Table 6.4</b> : Performance metrics for moderate demand data set.....  | 150         |
| <b>Table 6.5</b> : Performance metrics for high demand data set .....   | 150         |
| <b>Table 6.6</b> : Performance metric for actual field data. ....   | 151         |



## LIST OF FIGURES

|  | <u>Page</u> |
|--|-------------|
| <b>Figure 2.1</b> : One point crossover for binary representations. ....   | 36          |
| <b>Figure 2.2</b> : Two point crossover for binary representations. ....   | 36          |
| <b>Figure 2.3</b> : $N$ -point crossover ( $N = 3$ ) for binary representations. ....  | 36          |
| <b>Figure 2.4</b> : Uniform crossover for binary representations. ....   | 36          |
| <b>Figure 2.5</b> : Simple arithmetic recombination for real valued vectors. ....  | 37          |
| <b>Figure 2.6</b> : Single arithmetic recombination for real valued vectors. ....  | 37          |
| <b>Figure 2.7</b> : Whole arithmetic recombination for real valued vectors. ....   | 38          |
| <b>Figure 2.8</b> : Bitwise mutation for binary representations. ....  | 38          |
| <b>Figure 2.9</b> : Swap mutation for permutation representations. ....  | 39          |
| <b>Figure 2.10</b> : Insert mutation for permutation representations. ....   | 39          |
| <b>Figure 2.11</b> : Scramble mutation for permutation representations. ....   | 39          |
| <b>Figure 2.12</b> : Inversion mutation for permutation representations. ....  | 39          |
| <b>Figure 2.13</b> : Discrete recombination in evolutionary strategies. ....   | 41          |
| <b>Figure 2.14</b> : Intermediate recombination in evolutionary strategies. ....   | 42          |
| <b>Figure 2.15</b> : Representation of uncorrelated mutation with single step size. ....   | 44          |
| <b>Figure 2.16</b> : Representation of uncorrelated mutation with 2-step sizes. ....   | 45          |
| <b>Figure 2.17</b> : Representation of correlated mutation with 2-step sizes. ....   | 45          |
| <b>Figure 2.18</b> : $(\mu/\mu, \lambda)$ recombination in general scheme. ....  | 51          |
| <b>Figure 3.1</b> : Reflection to point $X_r$ . ....   | 56          |
| <b>Figure 3.2</b> : Extension to point $X_e$ from $X_r$ . ....   | 56          |
| <b>Figure 3.3</b> : Contraction points $X_c$ . (a) Outside contraction, (b) Inside contraction. ....   | 57          |
| <b>Figure 3.4</b> : Shrinking towards $X_l$ . Point $X_s$ and point $X_h$ comes closer to $X_l$ . ....   | 57          |
| <b>Figure 3.5</b> : Convergence graphs for BB–BC algorithms using best point as the<br>centre of mass (solid line) and the NM method (dashed line). .... | 58          |
| <b>Figure 3.6</b> : Generic algorithm flowchart. ....  | 63          |
| <b>Figure 3.7</b> : Local search phase for single step regression in BB-BC algorithm. ....   | 64          |
| <b>Figure 3.8</b> : Illustration of direction vector formation for local improvement with<br>single step regression in BB-BC Algorithm. ....             | 64          |
| <b>Figure 3.9</b> : Illustration of direction vector formation for local improvement with<br>double step regression in BB-BC Algorithm. ....             | 65          |
| <b>Figure 3.10</b> : Flowchart for the dichotomous local search algorithm. ....  | 66          |
| <b>Figure 3.11</b> : The hybrid BB-BC Algorithm versus the original BB-BC Algorithm. ....  | 67          |
| <b>Figure 3.12</b> : Improvement of the Ackley cost value. ....  | 68          |
| <b>Figure 3.13</b> : Improvement of the Ellipsoid cost value. ....   | 69          |
| <b>Figure 3.14</b> : Improvement of the Rastrigin cost value. ....   | 70          |
| <b>Figure 3.15</b> : Improvement of the Rosenbrock cost value. ....  | 71          |
| <b>Figure 3.16</b> : Improvement of the Sphere cost value. ....  | 72          |
| <b>Figure 3.17</b> : Improvement of the Rastrigin cost value. ....   | 73          |
| <b>Figure 3.18</b> : The improvements of the cost values for all functions for large search<br>space through evaluations. ....                           | 74          |

|  |     |
|--|-----|
| <b>Figure 3.19</b> : Benchmark test functions from CEC'05 competition. ....  | 78  |
| <b>Figure 4.1</b> : A sample gate allocation. ....   | 89  |
| <b>Figure 4.2</b> : A sample gate allocation – condensed view. ....  | 89  |
| <b>Figure 4.3</b> : Failure of greedy method. ....   | 91  |
| <b>Figure 4.4</b> : Illustration of failure of GTMA.....   | 91  |
| <b>Figure 4.5</b> : Algorithm flow.....  | 93  |
| <b>Figure 4.6</b> : Reordering of the flights: Type-a reordering. ....   | 94  |
| <b>Figure 4.7</b> : Reordering of the flights: Type-b reordering. ....   | 95  |
| <b>Figure 4.8</b> : Reordering of the flights: Type-c reordering. ....   | 95  |
| <b>Figure 4.9</b> : Graphical user interface for problem instance generation. ....   | 96  |
| <b>Figure 4.10</b> : Comparison of the three algorithms for moderate data set. ....  | 99  |
| <b>Figure 4.11</b> : Comparison of the three algorithms for high gate demand distributed<br>uniformly data set .....                             | 99  |
| <b>Figure 4.12</b> : Comparison of the three algorithms for high gate demand with demand<br>peaks data set.....                                  | 100 |
| <b>Figure 4.13</b> : Comparison of GTMA and SL-BBBC in a real world data set. ....   | 102 |
| <b>Figure 4.14</b> : The architecture for the resource management system.....  | 103 |
| <b>Figure 4.15</b> : An example gate allocation screen when flight information window is<br>open. ....   | 103 |
| <b>Figure 4.16</b> : An example gate allocation screen when manual edit window is open.<br>.....   | 104 |
| <b>Figure 4.17</b> : Operator display for gate allocating .....  | 104 |
| <b>Figure 4.18</b> : Comparison of the algorithms running on Atatürk Airport .....   | 107 |
| <b>Figure 5.1</b> : Mapping between genotype and phenotype space. ....   | 110 |
| <b>Figure 5.2</b> : Dominance rank.....  | 113 |
| <b>Figure 5.3</b> : Dominance count.....   | 114 |
| <b>Figure 5.4</b> : An illustration of fitness sharing. ....   | 115 |
| <b>Figure 5.5</b> : Another implementation of fitness sharing: Niching by gridding.....  | 116 |
| <b>Figure 5.6</b> : A relaxed dominance form .....   | 117 |
| <b>Figure 5.7</b> : Crowding distances.....  | 119 |
| <b>Figure 5.8</b> : Adaptive Gridding algorithm. ....  | 121 |
| <b>Figure 5.9</b> : Strength assignments of SPEA and SPEA-II.....  | 123 |
| <b>Figure 5.10</b> : Illustration of the archive truncation method used in SPEA2.....  | 124 |
| <b>Figure 5.11</b> : 2D illustration of two fictitious pareto front representations to be<br>compared. ....                                      | 128 |
| <b>Figure 6.1</b> : Main loop of the MOGAP algorithm.....  | 138 |
| <b>Figure 6.2</b> : Interchanging the order of N = 2 random flight pairs with random<br>distances away from random centers in the list. ....     | 139 |
| <b>Figure 6.3</b> : Algorithm progress.....  | 141 |
| <b>Figure 6.4</b> : The effect of archive postprocessing: (a) Pareto front shift, (b) Repairing<br>gaps, (c) Homogenizing the distribution. .... | 142 |
| <b>Figure 6.5</b> : The layout of a representative airport. ....   | 145 |
| <b>Figure 6.6</b> : The complete picture for the 30 independent pareto front representations<br>.....  | 152 |

# **ENHANCED HYBRID BIG BANG-BIG CRUNCH OPTIMIZATION ALGORITHMS AND APPLICATIONS ON SINGLE AND MULTI- OBJECTIVE AIRPORT GATE ASSIGNMENT PROBLEM**

## **SUMMARY**

Big Bang – Big Crunch (BB – BC) algorithm is a global optimization method relying on heuristics from the nature, particularly, the theory of Big Bang and Big Crunch. The algorithm generates new candidate solutions randomly in Big Bang phase and those solution candidates are latter used to obtain a single representative point through a contraction approach in the Big Crunch phase. One of the main contributions of this work is the local search hybridized version of the BB-BC, namely Big Bang-Big Crunch Algorithm with Local Directional Moves (BBBC-LS). The local search algorithm generates a direction vector by using the current representative point and the previous representative points of the generations and checks for improvement in that direction. If an improvement is achieved, the new centre is forced to switch to that point. That is to say, the centre point of the explosion of next big bang phase is changed. The step size of the local search is set and adjusted according to the distance between these consecutive representative points. The exploitation or intensification capability of the algorithm is enhanced with local search; and thus, the proposed hybridization operation produces much more accurate results than the original BB – BC algorithm. In fact, it also provides promising results when compared to the state-of-the-art optimization methods. Moreover, the newly proposed algorithm is shown to be much more effective in terms of complexity.

Airline industry has been using operation research techniques for more than fifty years. In the last three decades, rapid developments in the computational powers of the processors paved the way for utilizing highly complex planning and scheduling strategies. Both the airlines and airport operators make use of problem tailored algorithms to maximize their revenues. One of the most important limitations in the resources of an airport is in the allocation of gates to the planes; and consequently, gate assignment plays a major role in the revenue obtained from ground operations.

Gate Assignment Problem (GAP) is well studied in the literature and consequently, there are many proposed problem formulations and solution techniques. Though the basic constraints and objectives are easily perceived, the problem has many interactions with other resources such as the number of gates, airport topology, flight schedules, distances to baggage claim areas, etc. Therefore, GAPs are even more complicated than most other traditional scheduling problems. Moreover, as the air traffic becomes more demanding, the grandeur of the solution space gets even larger; in return, this makes traditional binary integer techniques practically inapplicable. In those cases, nature inspired computing techniques became a good alternative for GAPs.

One other main contribution of the study is the GAP solution techniques proposed for both single and multi-objective gate assignment problems. The solution approaches combine the benefits of heuristic approaches that provide a fast initiating solution to the problem and later conduct stochastic searches in order to ameliorate the previously obtained result via heuristic approaches. The solution techniques are Single Leap-Big Bang Big Crunch (SL-BBBC) algorithm for single objective problems and enhanced Order Based Single Leap-Big Bang Big Crunch (eSL-BBBC) algorithm for multi-objective problems. The algorithms are experimented on various artificial and actual field data to illustrate performance.

The main contributions of the study can be listed as follows:

- a. Proposing Big Bang-Big Crunch with Local Directional Moves (BBBC-LS) algorithm that possesses improvements over algorithmic capability of the classical Big Bang-Big Crunch (BB-BC) optimization method. The effectiveness of the hybridized algorithm has been illustrated on various test beds.
- b. Investigating Airport Gate Assignment Problem (AGAP) and proposing practically applicable problem formulations.
- c. Introducing Single Leap-Big Bang Big Crunch (SL-BBBC) algorithm for the solution of single objective AGAP.
- d. Proposing a systematic method for parameter-controlled quasi-realistic airport data generation for quasi-real simulations.
- e. Discussing previous work on multi-objective airport gate assignment problem and proposing a state-of-the-art solution strategy named as enhanced Order Based Single Leap-Big Bang Big Crunch (eSL-BBBC) method.



# GELİŞTİRİLMİŞ MELEZ BÜYÜK PATLAMA-BÜYÜK ÇÖKÜŞ OPTİMİZASYON ALGORİTMALARI VE TEK VE ÇOK AMAÇLI HAVALANI KAPI ATAMA PROBLEMİ UYGULAMALARI

## ÖZET

Büyük Patlama-Büyük Çöküş eniyileme algoritması, evrenin oluşumunu açıklayan en önemli teorilerden Büyük Patlama ve Büyük Çöküş teorilerine dayanan bir global eniyileme yöntemidir. Büyük Patlama-Büyük Çöküş algoritması ile çözüm adaylarından oluşan bir toplulukta, nesiller (iterasyonlar) boyunca değişime uğrayan bireylerin problemin çözümüne yakınsaması sağlanır. Bireylerin değişime uğraması, genetik algoritmalarındaki mutasyon operatörünün işlevine yakın bir şekilde büyük patlama ve büyük çöküş fazları ile sağlanır. Büyük patlama, belli bir nokta etrafında standart sapması kontrol edilen bir normal dağılım vektörünün parametre uzayında bireylere eklenmesi işlemidir. Büyük çöküş operatörü ise arama uzayına dağılmış bireylerin ortak olarak belirlediği bir çökme noktasının ilgili iterasyonun sonucunda elde edilen en iyi nokta, yani ilgili iterasyonunun çözümü olarak hesaplanması işlemidir. En basit haliyle büyük çöküş operatörü arama uzayında en iyi amaç değere sahip bireyin yeri olarak belirlenebilir. Bununla birlikte en verimli yöntem her bireyin çökme noktasına katkıda bulunduğu ağırlıklı ortalama yöntemidir. Buna göre her birey amaç değerinin büyüklüğüne / küçüklüğüne göre ağırlıklandırılarak ortalama alınır ve ilgili iterasyonun çözümü olarak sunulur.

Bu çalışmanın en önemli katkılarından birisi de Yerel Yönsel İlerlemeli Büyük Patlama Büyük Çöküş isimli yöntemin geliştirilmiş olmasıdır. Yerel arama yöntemlerinin evrimsel aramalar içinde kullanılması ve melez yapılar oluşturularak her iki yaklaşımın güçlü yönlerinin uygun şekilde değişmeli olarak kullanılması literatürde sıkça rastlanan bir durumdur. Yerel Yönsel İlerlemeli Büyük Patlama Büyük Çöküş eniyileme algoritmasında yerel arama algoritması güncel nesillerinin çözümü (jenerasyon merkezi, bir sonraki patlama merkezi) ile önceki nesillerin çözümü arasında oluşturulan yönde özelleşmiş arama yapmaktan sorumludur. Bu arama için, basit yerel arama yöntemlerinden bölerek arama ya da ikircilli arama kullanılır. Üretilecek arama vektörü ya da arama alanı sadece bir önceki ya da bir ve iki önceki nesil çözümleri kullanılarak belirlenir. Yerel arama adımları ile bir iyileşme sağlanırsa nesilin çözüm değeri; başka bir ifadeyle bir sonraki Büyük Patlama'nın merkezi bu noktaya taşınır. Yerel arama yöntemi, iterasyonlar arası elde edilen çözümlerin birbirlerine uzaklığına dayalı olarak arama alanının büyüklüğünü değiştireceğinden, kendi kendini uyarlayabilir yapıdadır. Böylece ilk iterasyonlarda büyük bir alanda daha az yoğun bir arama icra edilirken; algoritmanın son iterasyonlarının arasındaki yerel aramalar çözüm üzerinde ince ayar yapmaktadır. Yerel arama işlevi, iterasyonlar arasına, Büyük Patlama-Büyük Çöküş algoritmasının adımlarına müdahale etmeden eklenmiştir.

Yerel Yönsel İlerlemeli Büyük Patlama-Büyük Çöküş yönteminin bir diğer özelliği Büyük Çöküş operatörü olarak Nelder-Mead eniyileme yöntemini kullanabilmesidir.

Literatürde sıklıkla kullanılmış olan bu yöntem, Büyük Çöküş operatörü olarak en iyi seçme ve ağırlıklı ortalamaya yöntemlerine bir alternatif olmaktadır. Hangi Büyük Çöküş operatörünün kullanılacağı algoritma koşumu esnasında önceden belirlenen parametre ile kontrol edilerek değiştirilebilmektedir. Bu sayede, arama başlangıcındaki topoloji bilgisinin az olduğu iterasyonlarda diğer Büyük Çöküş operatörleri ile daha hızlı yakınsama sağlanıp, arama olgunlaştıktan sonra Nelder-Mead çokgenleri ile çözüm doğruluğu artırılabilir.

Yerel Yönelimli İlerlemeli Büyük Patlama-Büyük Çöküş eniyileme algoritması ile elde edilen sonuçların Büyük Patlama-Büyük Çöküş eniyileme algoritması ile elde edilen sonuçlara göre oldukça iyileştiği; buna karşın algoritmanın harcadığı süre ve karmaşıklığının ihmal edilebilir oranda arttığı benzetim sonuçları ile gösterilmiştir. Dünya çapında yaygın kabul görmüş eniyileme yöntemleri (Genetik Algoritmalar, Evrimsel Stratejiler, Parçacık Sürü Optimizasyonu) ile karşılaştırıldığında, önerilen yöntemin doğruluk ve hız açısından üstün sonuçlar verebildiği; karmaşıklık metriklerinde ise çok daha üstün olduğu gösterilmiştir.

Havaalanlarında ve havayolu işletmelerinde eniyileme yöntemlerinin birçok kullanım alanı vardır. Son otuz yılda bilgisayar işlemcilerinin güçlerindeki hızlı artış, çok karmaşık planlama ve çizelgeleme yöntemlerinin gerçekleştirilebilmesine olanak sağlamıştır. Her havayolu şirketleri hem de havaalanı işletme şirketleri bu alanda yatırımlar yaparak kazançlarını artırma yoluna gitmişlerdir. Havaalanlarındaki en önemli kısıtlı kaynaklardan biri de uçakların yanaştığı kapılardır. Dolayısıyla kapıların artan hava trafiğine en verimli şekilde hizmet etmesi gerekmektedir.

Havaalanı kapı atama problemi literatürde ve pratik uygulamalarda özellikle son on yılda çokça çalışılmış ve birçok matematiksel problem tanımı ve çözüm tekniği önerilmiştir. Temel kısıtlar ve amaç fonksiyonları kolayca anlaşılabilir olmasına rağmen hava alanı kapı atama problemi, kapı sayısı, havaalanı topolojisi, uçuş planları, havaalanı içindeki yürüme mesafeleri gibi dış etkenlere bağımlılığı dolayısıyla karmaşık bir problemdir ve NP-zor olarak sınıflandırılır. Son yıllarda hızla artan havaalanı trafiği, problemin bir tamsayı problemi olarak klasik yöntemlerle çözülebilmesini zorlaştırmıştır. Bu nedenle evrimsel arama yöntemlerinin kapı atama problemine uygulanması yeni ortaya çıkan ve pratikte kullanım alanı bulan bir konudur.

Bu çalışmanın bir diğer ana katkısı, havaalanı kapı atama problemine yeni bir matematiksel problem tanımı getirmek, bu tanım çerçevesindeki tek ve çok amaç fonksiyonuna sahip problemleri çözebilmek için evrimsel hesaplama dayalı yöntemler önermektir. Önerilen çözüm yöntemleri, deterministik sezgisel yaklaşımlar sayesinde ilk çözümü hızlı bir şekilde oluşturduktan sonra stokastik evrimsel yöntemlerle bu çözümü iyileştirmeyi amaçlar.

Tek amaç fonksiyonu ile tanımlanan kapı atama probleminde amaç, kapılarda uçakların kalış süresini en çoklamaktır. Bir diğer ifade ile amaç, aprona çekilmek zorunda kalınan uçakların havaalanında kalacağı toplam sürenin en azlanmasıdır. Bu amaç fonksiyonunun en azlanması için öncelikle uçakları kalış sürelerinin çokluğuna göre sıralayan sezgisel yaklaşımla uçakların sıralanması sağlanır. Bu sıralama, iterasyonlar boyunca değiştirilip uçak yerleşimleri kontrol edilerek çözüme ulaşılır. Uçakların yerleştirilmesi işlemi sıralama kavramı ile ilişkilendiren bu yöntemin etkinliği üretilen yapay veri kümelerinde, İstanbul Atatürk Havaalanından elde edilen veriler üzerinde gösterilmiştir. Ayrıca geliştirilen algoritma TAV Bilişim A.Ş.

firmasının havaalanı kaynak planlama sistemi içerisinde gerçekleşmiş ve ticari bir ürün olarak çeşitli havaalanlarında kullanımı sağlanmıştır.

Çok amaçlı havaalanı kapı atama probleminde amaçlar; kapıların doluluğunu en çoklamak, yolcu yürüme mesafelerinin toplamını en aza indirmek ve çeşitli kriterlerin birleşimi olan önceliklerin karşılanmasını en çoklamak olarak tanımlanmıştır. Bu amaç fonksiyonları yer yer birbirleriyle çeliştiğinden tümünü birden eniyileyen bir çözüm bulmak her zaman mümkün değildir. Bu nedenle çözüm, en az bir amaç fonksiyonu açısından diğerlerinden iyi olan bireylerden oluşan kümedir. Pareto optimum kümesi olarak adlandırılan bu kümenin, çok amaçlı hava alanı kapı atama problemi için, Genişletilmiş Uçak Sıralaması Tabanlı Tek Atlamalı Büyük Patlama-Büyük Çöküş Yöntemi ile daha önce literatürde önerilen yöntemle göre daha iyi şekilde oluşturulduğu çeşitli metrikler baz alınarak gösterilmiştir.

Havaalanı kapı atama algoritmalarının testlerinde veri kümesinin gerçeğe yakınlığı, havaalanlarında kullanılabilir algoritmalar üretebilmek için çok önemlidir. Uçakların ortalama kalış süreleri, kalış sürelerinin standart sapması, yürüme mesafeleri, yolcu sayıları ve öncelikleri belirleyen modellerin dikkatle oluşturulması gerekir. Bu çalışmada, havaalanı kapı atama algoritmalarının testleri için parametrik olarak kontrol edilebilen gerçekçi bir veri üretici tasarlanmıştır. Bu veri üretici ile elde edilen algoritma sonuçları, İstanbul Atatürk Havaalanından elde edilen verilerle yapılan deney sonuçları ile yüksek benzerlik göstermiştir.

Tez çalışmasının içeriği aşağıdaki gibi maddelenebilir:

- a. Sürekli eniyileme problemlerinin çözümüne yönelik Yerel Yönsel İlerlemeli Büyük Patlama-Büyük Çöküş algoritması önerilmiştir. Algoritmanın doğruluk, hız ve karmaşıklık analizi hem Büyük Patlama-Büyük Çöküş yöntemi hem de literatürde en çok kabul görmüş yöntemlerle karşılaştırmalı olarak verilmiştir.
- b. Havaalanı kapı atama problemi için sahada gerçekleştirilebilir problem tanımlamaları önerilmiştir.
- c. Tek amaç fonksiyonuna sahip havaalanı kapı atama problemi için Tek Atlamalı Büyük Patlama-Büyük Çöküş yöntemi önerilmiş ve algoritma etkinliği çeşitli test kümelerinde gösterilmiştir. Önerilen yöntemin farklı amaç fonksiyonları üzerinde kullanılabilirliği tartışılmıştır.
- d. Uçuş planlaması ve havaalanı yolcu trafiği için parametrik yönetilebilen gerçeğe yakın bir test verisi üretici sunulmuştur. Geliştirilen algoritmaların benzetimlerinde, yapay üretilen bu verilerin sahadan toplanan verilerle uyumluluğu ortaya konmuştur.
- e. Çok amaçlı kapı atama problemine ilişkin geçmişte raporlanan çalışma detayları ile incelenmiş, çalışmanın zayıflıkları ortaya konarak daha etkili bir yöntem olan Genişletilmiş Uçak Sıralaması Tabanlı Tek Atlamalı Büyük Patlama-Büyük Çöküş Yöntemi önerilmiştir. Önerilen yöntem, çeşitli yapay test kümeleri üzerinde ve İstanbul Atatürk Havaalanı'ndan elde edilen gerçek saha verileri üzerinde test edilerek etkinliği incelenmiştir.



## 1. INTRODUCTION

### 1.1 What is an Evolutionary Algorithm?

Every real-world problem from economic to scientific and engineering fields is ultimately confronted with a common task, optimization. An optimization problem can be defined by specifying the set of all feasible candidates and a measure for evaluating their worth (Ahn, 2006).

As the result of intense research over the years, there are many optimization algorithms reported. One of the main classes of optimization algorithms is the evolutionary algorithms.

Evolutionary algorithms are the umbrella term for many stochastically developed population based search techniques that are inspired from the natural evolution process. The analogy in between the natural evolution process and the optimization problem is given in **Table 1.1**.

**Table 1.1:** Evolutionary computing metaphor.

| Natural Evolution | Optimization<br>Problem Solving |
|-------------------|---------------------------------|
| Environment       | Problem                         |
| Individual        | Candidate Solution              |
| Fitness           | Quality                         |

Frequently, Evolutionary Computation, Evolutionary Optimization, Evolutionary and Programming terms are interchangeably used. The slight differences of these terms are ignored in this thesis.

Evolutionary computing techniques are based on Mendelian Genetics and Darwinian Theory of Evolution. They, somehow imitate the nature to find out what is best for some specific problem. In today's world, they have been successfully applied to

many areas such as scheduling applications, system design, learning and prediction applications, automated program development, multi-criteria decision-making, evolvable hardware design, etc.

There are many variants of the evolutionary algorithms. Nevertheless, the common underlying idea behind all these techniques is the same: given a population of individuals, the environmental pressure causes natural selection that is survival of the fittest, which causes a rise of the fitness of the population. Given a quality function to be maximized, a set of candidate solutions can be created randomly, then these solutions can be scored by applying the quality function as an abstract fitness measure. Based on this fitness scores, some of the better candidates are chosen to seed the next generation by applying recombination and/or mutation (Eiben and Smith, 2003).

## **1.2 A Brief History on Evolutionary Algorithms**

History of the evolutionary computation gets back to Turing when he first proposed genetical or evolutionary search concepts in 1948. Then in 1962, Bremermann executed a computer program on optimization through evolution and recombination (Fogel, 1998). After the 60s, when the evolutionary computation concepts accelerated to grow, three main branches emerged including Evolutionary Programming (Fogel et al, 1965; Fogel et al, 1966), Genetic Algorithm (De Jong, 1975; Holland, 1973; Holland, 1975) and Evolutionary Strategies (Rechenberg, 1973; Schwefel, 1995). Up to the 90s, these works are interpreted as separate fields of research, but now, as stated in the previous chapter they are classified under the term “evolutionary computation”. In 90s, also a fourth branch following the same concepts with a different approach emerged and became the final main branch of evolutionary computation: genetic programming (Banzhaf et al, 1998; Koza, 1992; Koza 1994).

For more detailed literature survey, one can investigate Fogel (1998) and De Jong (2006).

### **1.3 Airport Gate Assignment Problem**

The air transportation becomes more and more widespread during the past fifteen years. As well as the opportunity of travelling long distances in reasonable short time duration, the moderate prices due to competition of the companies made several travelers to choose airline industry. These facts tremendously increased the traffic in the airports compared to mid-1990s. Assigning arriving flights to airport gates is an important issue in daily operations of an airline. It has a major impact on maintaining the efficiency of flight schedules, passenger satisfaction and the revenue obtained. Gate assignment problem is a quadratic assignment problem and the solution algorithm should handle large search spaces. Therefore, the evolutionary optimization algorithms can be good solution alternatives.

### **1.4 Purpose of Thesis**

This thesis has the following purposes,

- I. To propose improvements on algorithm capability of the Big Bang-Big Crunch (BB-BC) optimization algorithm in numeric problem domains and illustrate the improvements on various test beds.
- II. To investigate airport gate assignment problem (AGAP) and propose practically applicable problem formulations.
- III. To propose an evolutionary method on the solution of AGAP
- IV. To generate quasi-realistic airport data for real-like simulations
- V. To discuss previous work on multi-objective gate assignment problem (MOGAP) and propose a state-of-the-art solution strategy remedying the weaknesses.

### **1.5 A Brief Summary of Chapters**

Chapter 2 discusses the Big Bang-Big Crunch (BB-BC) algorithm within the evolutionary computation methods. In this context, the chapter briefly reviews the Genetic Algorithms (GA), Evolutionary Strategies (ES) and Big Bang-Big Crunch (BB-BC) algorithm. In the final subchapter, relations and differences of BB-BC with the previously reported literature is investigated.

In chapter 3, a new method based on BB-BC is introduced: Big Bang-Big Crunch Algorithm with Local Directional Moves (BBBC-LS). The method is shown to be good alternative for the well-accepted methods as Genetic Algorithms, Evolutionary Strategies and Particle Swarm Optimization.

Chapter 4 introduces the total time slot maximization formulation for the AGAP. The problem is solved by the Single Leap-Big Bang Big Crunch (SL-BBBC), which is one of the main contributions of the study. In this chapter, the practical application of the problem is given.

Chapter 5 briefly reviews the basics of multi-objective optimization. Performance metrics used in this work are also given.

Chapter 6 introduces the Enhanced Order Based Single Leap-Big Bang Big Crunch (eSL-BBBC) optimization algorithm on the solution of multi-objective gate assignment problems (MOGAPs). In this chapter, a test data generator for quasi-realistic airport flight data and airport pedestrian traffic data is introduced.

Finally, chapter 7 gives some conclusions and further recommendations.



## **2. BIG BANG-BIG CRUNCH ALGORITHM WITHIN EVOLUTIONARY COMPUTATION METHODS**

### **2.1 Components of the Evolutionary Algorithms**

All the evolutionary algorithms have a number of components in common. These can be listed in a generic manner as,

- I. Representation
- II. Objective Function
- III. Population
- IV. Parent Selection
- V. Variation Operators
- VI. Survivor Selection

Here only the basic aspects for the terms are given.

#### **2.1.1 Representation**

The initial step of constructing the evolutionary algorithm is defining the mapping between the original problem space (phenotypes) and the problem solving space (genotypes). With respect to the nature of the problem, the parameters to be tuned are encoded in the genotype. The variation operators also act on genotype. Then, the results are mapped into their corresponding phenotypes for fitness evaluation. Representation of the solutions includes the selection of the genotypic expression (like binary coding, integer or floating representations and permutation representations) and encoding them into phenotypes.

#### **2.1.2 Objective function**

Objective function (also named as fitness function or cost function) is the component that defines the problem. The evolution trend is through to the global minimum (or maximum) of the objective function. It determines how well the particular candidate

solution is by assigning it a score value. These scores are used for mating pool selection in the next phase.

### **2.1.3 Population**

The population includes the full set of candidate solutions at each generation. Working with population of solutions provides an environment to simulate natural survival of the fittest process.

The diversity of a population is a measure of the number of different solutions present. No single measure for diversity exists. Typically, people might refer to the number of different fitness values present, the number of different phenotypes present, or the number of different genotypes. Other statistical measures such as entropy are also used. Note that only one fitness value does not necessarily imply only one phenotype is present, and in turn, only one phenotype does not necessarily imply only one genotype. The reverse is, however, not true: one genotype implies only one phenotype and fitness value (Eiben and Smith, 2003).

### **2.1.4 Parent selection**

Parent selection (mating selection) is the process of selecting the parents for the next generation. The parents are selected with respect to the fitness scores assigned. Then they undergo some changes by the variation operators to produce children.

### **2.1.5 Variation operators**

Variation operators produce new individuals from the mating pool parents. Designing a variation operator is the key point on designing an evolutionary algorithm. Variation operators can work on single parent (asexual reproduction) or two parents (sexual production). There are also multi-parent variation operators reported in the literature.

Most commonly accepted name for asexual reproduction operators is mutation. It causes a random, unbiased change in the genotype of the parent individual. On the other hand, binary variation operators are generally referred as recombination or crossover operators. These operators work on two parent genotypes to produce one child or two children.

### **2.1.6 Survivor selection**

Survivor selection (often referred to as replacement or environment selection) is the process of selecting the individuals for the next generation. In most widespread approaches for survivor selection, either the children will all survive or some “fit” parents will still be in the population of the next generation.

As opposed to parent selection, which is typically stochastic, survivor selection is often deterministic, for instance, ranking the unified multiset of parents and offspring and selecting the top segment (fitness biased), or selecting only from the offspring (age biased) (Eiben and Smith, 2003).

## **2.2 Algorithms Reported in the Literature**

There are many evolutionary computation algorithm variants reported on the literature. In this thesis, a comprehensive introduction for Genetic algorithms, Evolution Strategies and Big Bang-Big Crunch Method is given. Genetic algorithms have been accepted in a wide sense and the most known variant for EAs. On the other hand, Evolution Strategies have many similar aspects with the Big Bang-Big Crunch Algorithm that is in the focus of the dissertation.

### **2.2.1 Genetic algorithms**

Genetic algorithms (GAs) are the most known evolutionary algorithm variants. There is no single formulation for the genetic algorithm development; instead, it is tailored for the specific problem.

#### **2.2.1.1 Representation in GA**

There are four basic representations for the individuals for GAs:

- I. Binary representation,
- II. Integer representation,
- III. Real valued or floating-point representation and
- IV. Permutation representations.

Unfortunately, the designer can select the best representation for a specific problem only by experience; there is no systematic way. The selection of the representation

directly effects the variation operators used and, of course, the encoding of the genotype into phenotype. The encoding should map all possible genotype combinations to valid phenotypes.

### **2.2.1.2 Population in GA**

Generally, the population is initialized randomly. The population of each generation should be diverse enough to yield new populations through variation operators. In the most common sense, one can classify the population models into two classes: steady state and generational. The more frequently used model is generational population model and in that one all the members of the population are replaced by the children formed after processing of the variation operators. In widely accepted notation,  $\mu$  designates the number of parents selected for the mating pool and  $\lambda$  designates the number of children (offspring) created. Then, in generational models  $\lambda = \mu$ . On the other hand in steady state population models not all of the individuals are replaced, instead, some of the offspring are selected (generally  $\lambda < \mu$ ) and injected in the next generation. Parent replacement is done based on ages or fitness scores of the members.

Selecting the size of the population is quite a fuzzy concept and generally depends on experience or trial and errors. For detailed investigation, one can investigate Goldberg et al. (1992).

### **2.2.1.3 Parent selection in GA**

Parent selection for mating pool is performed in the favor of better members in all parent selection algorithms. However, this bias should not prevent the population to preserve its diversity. There are three commonly accepted modes for the parent selection,

- I. Fitness Proportional Selection,
- II. Ranking Selection and
- III. Tournament Selection.

Fitness Proportional Selection (FPS) (Holland, 1975) assigns probabilities for each individual with respect to their absolute fitness.

In Ranking Selection (RS) (Baker, 1987), the individuals are sorted with respect to fitness scores and the probabilities are assigned respecting this order.

FPS and RS techniques are stochastic methods assigning a probability value for each individual. Mapping these probabilities to actual selection counts is the next step. There are two prominent algorithms: Roulette Wheel Selection (RWS) and Stochastic Universal Sampling (SUS) (Baker, 1987).

If the population size is considerably large, calculating all the fitness scores and sorting them can be very expensive. Instead, the members can be raced in subsets and the winner goes to the mating pool. This method is named as tournament selection since a tournament is organized among a subset of individuals and the winner is prized.

#### **2.2.1.4 Variation operators in GA**

In this subchapter, unary (mutation) and binary (crossover, recombination) variation operators are discussed. The algorithms for the variation operators are heavily dependent on representations of the individuals in the population.

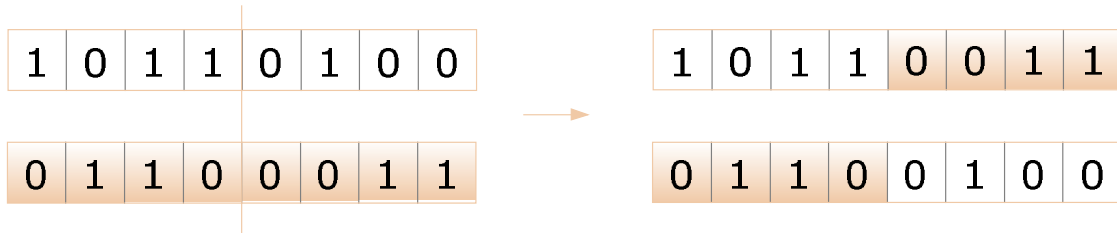
#### **2.2.1.5 Recombination operators**

In common approach, binary variation operators produce two children from two parents. One other important aspect for the crossover operators is to ability to inherit common genes to the offspring (Radcliffe, 1991). The whole set of operators listed here have this property accept for partially mapped crossover for permutation representations.

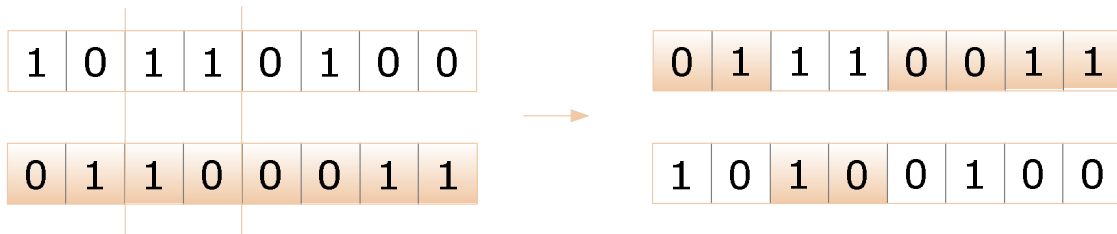
For binary representations, the basic methods of recombination are,

- I. one point crossover (Holland, 1975; De Jong, 1975), (**Figure 2.1**)
- II. two point crossover, (**Figure 2.2**)
- III.  $N$ -point crossover (**Figure 2.3**) and
- IV. uniform crossover. (**Figure 2.4**)

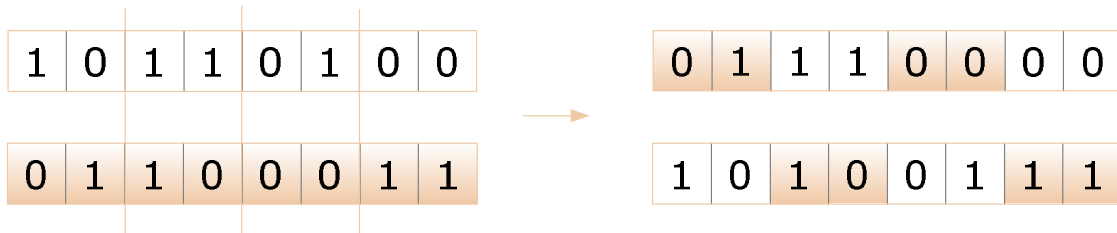
The first three crossover types have tendency to take neighboring genes together, a phenomena named positional bias (Eshelanian et al 1989; Spears and De Jong, 1991). On the other hand, uniform crossover has the distributional bias since it is expected to transmit equal number of genes from both parents from random positions.



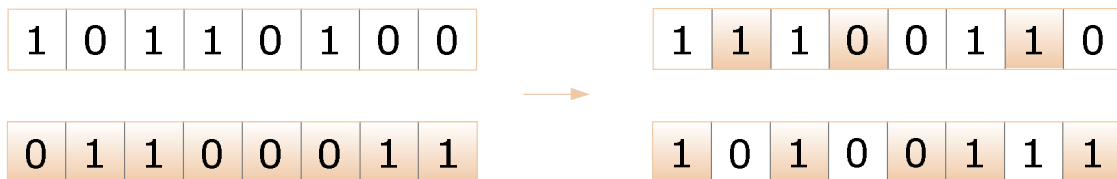
**Figure 2.1 :** One point crossover for binary representations.



**Figure 2.2 :** Two point crossover for binary representations.



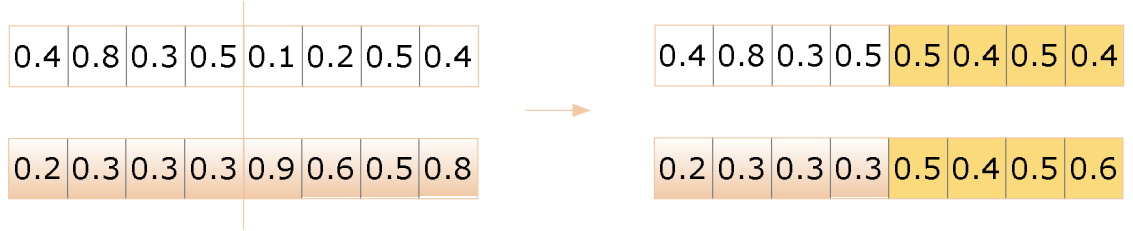
**Figure 2.3 :**  $N$ -point crossover ( $N = 3$ ) for binary representations.



**Figure 2.4 :** Uniform crossover for binary representations. Eight random numbers are drawn for each gene and the 2<sup>nd</sup>, 4<sup>th</sup> and the 7<sup>th</sup> numbers are above 0.5.

For integer and floating point representations the same set of operators with that of binary representations are used. For floating point representations, also arithmetic recombination operators can be defined. Simple arithmetic recombination proposes to choose a crossover point  $P$ . Then take first  $P$  genes from parent one, and then the other genes are weighted average of the two parents. In the second child, first  $P$  genes are taken from the second parent and the remaining genes are again a weighted mean of the parents (**Figure 2.5**). The mathematical model for simple arithmetic recombination is (2.1),

$$\begin{aligned}
& \text{Parent - 1} : \langle a_1, a_2, \dots, a_n, a_{n+1}, \dots, a_l \rangle \\
& \text{Parent - 2} : \langle b_1, b_2, \dots, b_n, b_{n+1}, \dots, b_l \rangle \\
& \text{Offspring - 1} : \langle a_1, a_2, \dots, a_n, \alpha b_{n+1} + (1 - \alpha) a_{n+1}, \dots, \alpha b_l + (1 - \alpha) a_l \rangle \\
& \text{Offspring - 2} : \langle b_1, b_2, \dots, b_n, \alpha a_{n+1} + (1 - \alpha) b_{n+1}, \dots, \alpha a_l + (1 - \alpha) b_l \rangle
\end{aligned} \tag{2.1}$$



**Figure 2.5** : Simple arithmetic recombination for real valued vectors. ( $P = 4$ ,  $\alpha = 0.5$ ).

Single arithmetic recombination and whole arithmetic recombination can be illustrated in **Figure 2.6** and **Figure 2.7** and in (2.2).

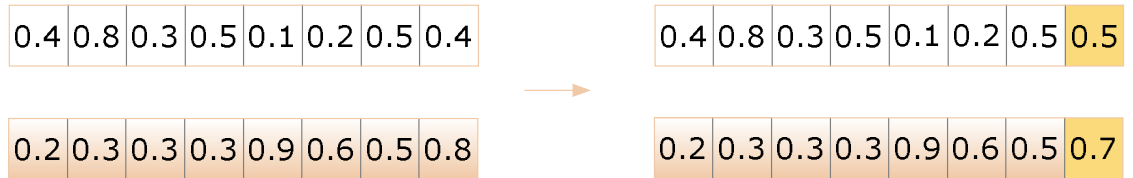
$$\begin{aligned}
& \text{Parent - 1} : \langle a_1, a_2, \dots, a_n, a_{n+1}, \dots, a_l \rangle \\
& \text{Parent - 2} : \langle b_1, b_2, \dots, b_n, b_{n+1}, \dots, b_l \rangle
\end{aligned}$$

SingleArithmeticRecombination:

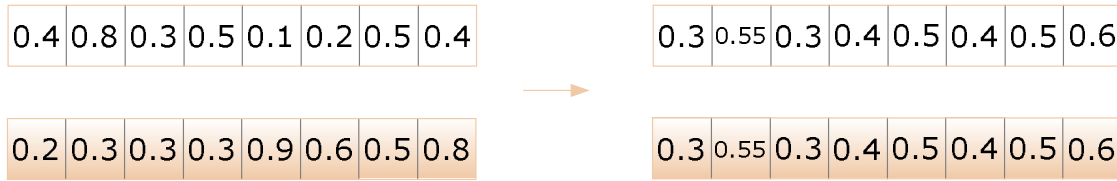
$$\begin{aligned}
& \text{Offspring - 1} : \langle a_1, a_2, \dots, a_n, \alpha b_{n+1} + (1 - \alpha) a_{n+1}, \dots, a_l \rangle \\
& \text{Offspring - 2} : \langle b_1, b_2, \dots, b_n, \alpha a_{n+1} + (1 - \alpha) b_{n+1}, \dots, b_l \rangle
\end{aligned}$$

WholeArithmeticRecombination:

$$\begin{aligned}
& \text{Offspring - 1} : \langle \alpha b_1 + (1 - \alpha) a_1, \alpha b_2 + (1 - \alpha) a_2, \dots, \alpha b_{n+1} + \\
& \quad (1 - \alpha) a_n, \alpha b_n + (1 - \alpha) a_{n+1}, \dots, \alpha b_l + (1 - \alpha) a_l \rangle \\
& \text{Offspring - 2} : \langle \alpha a_1 + (1 - \alpha) b_1, \alpha a_2 + (1 - \alpha) b_2, \dots, \alpha a_n + \\
& \quad (1 - \alpha) b_n, \alpha a_{n+1} + (1 - \alpha) b_{n+1}, \dots, \alpha a_l + (1 - \alpha) b_l \rangle
\end{aligned} \tag{2.2}$$



**Figure 2.6** : Single arithmetic recombination for real valued vectors. ( $n = 7$ ,  $\alpha = 0.25$ ).



**Figure 2.7 :** Whole arithmetic recombination for real valued vectors. ( $\alpha = 0.5$ ).

Crossover operator design is hard for permutation problems since a simple exchanging operation arouses multiple copies of parameters in the chromosomes. Partially Mapped Crossover (Goldberg and Lingle, 1985) was proposed for adjacency-based problems and the algorithm run can be investigated from Whitley (2000). Edge crossover, order crossover (Davis, 1991) and cycle crossover (Oliver et al, 1987) are other well-applied permutation crossovers designed for order-based representations.

### 2.2.1.6 Mutation operators

For binary representations, mutation is performed on every bit with a small probability (**Figure 2.8**). Selection of the probability depends on the problem but in common sense, the expected value of the mutant bit number is 1. Then the probability of mutation is selected to be  $1 / (\text{length of the chromosome})$ .



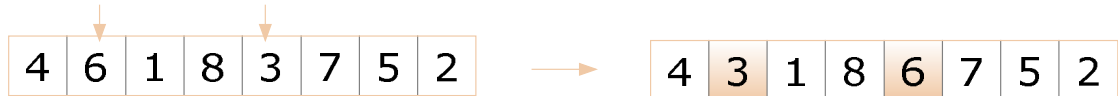
**Figure 2.8 :** Bitwise mutation for binary representations.

In integer representation, in connection with the binary mutation, random resetting draws a random number from the permissible set of integers. Random resetting is suitable for cardinal attributes. For ordinal attributes, creep mutation can be used in which a small value is added to each gene. The value added is drawn from a normal distribution with mean zero and a small variance.

Uniform mutation for floating point representations replaces a certain parameter value by a randomly drawn new one in the permissible interval. This mutation is analogous of the bit flipping of binary representations and random resetting of the integer representations. For floating point representations, the analogous of the creep mutation is the non-uniform mutation (with a fixed distribution such as Gaussian or Cauchy distribution) (Michalewicz, 1992).



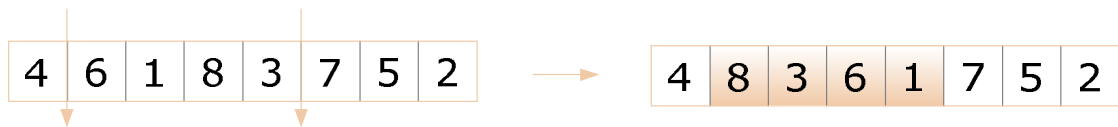
Mutation operators designed for the permutation representations differ from the others since they cannot work on each gene separately. In swap mutation, the randomly selected two genes are swapped (**Figure 2.9**). In insert mutation, a randomly selected gene is transported to another randomly selected one moving the others (**Figure 2.10**). In the scramble mutation, between the randomly selected two genes, all the genes are reordered (**Figure 2.11**), and as a specific case, inversion mutation proposes to inversely reordering this subset (**Figure 2.12**).



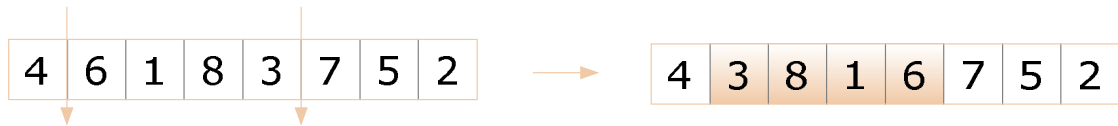
**Figure 2.9 :** Swap mutation for permutation representations.



**Figure 2.10 :** Insert mutation for permutation representations.



**Figure 2.11 :** Scramble mutation for permutation representations.



**Figure 2.12 :** Inversion mutation for permutation representations.

### 2.2.1.7 Survivor selection in GA

Survivor selection or replacement concept is quite connected with the population model. If a steady population model is selected, generally the number of offspring is less than the population size. There are age based and fitness based methods for selecting which offspring and current members will survive to the next generation. In age based methods, a FIFO model or a stochastic selection where the probability of selection decreases with increasing age can be implemented. The replaced portion of the population is referred as generational gap.

One another important point to note is the elitism strategy. Not to lose the current best members by age based or stochastic fitness based replacement strategies, best

$n$  members of the population can be protected from elimination. Number  $n$  is commonly selected as one or two.

### **2.2.2 Evolution strategies**

The main distinction of the Evolution Strategies (ES) from Genetic Algorithms is the inherited self-adaptation concept. In the broadest sense, self-adaptation is the dynamic altering of the algorithm run parameters throughout iterations. This feature is generally provided by including the parameters governing the algorithm run into the chromosome structure and therefore allowing them to co-evolve with the solutions. The famous rule of 1/5 success rule (Rechenberg, 1973) is one of the most known and used adaptation rule for controlling mutation step size. This rule states that if the ratio of the successful mutations (that is mutations yielding a fitter member) to all mutations should be 1/5. Therefore, if it is above this threshold, the mutation step size is increased; else it should be decreased. This check is performed at some specific period of iterations. Note that this adaptation process do not interacts with the chromosome representation and actually not a state-of-the-art technique.

#### **2.2.2.1 Representation in ES**

Evolution strategies are used in continuous optimization problems, hence real valued representation is used. Genotype and phenotype are usually the same; therefore, there is no need for a special encoding scheme. What is new for ES is the inclusion of control parameters (= strategy parameters) in the chromosome structure. The strategy parameters are, for the common sense, divided into two sets. One set is the parameters for mutation step size control ( $\sigma$ ), the other set is the parameters for controlling the dependencies of the step sizes of the different parameters (covariance of  $\sigma$  set,  $C_\sigma$ ).  $\sigma$  set must have at least one member valid for all elements of the chromosome for self adaptation implementation. Generally,  $\sigma$  set has either 1 or  $N$  elements,  $N$  being the dimension of the problem.  $C_\sigma$  set is not always used, but is needed for non-symmetrical mutation effects that will be later.  $C_\sigma$  set has different number of elements.

The chromosome structure is -in the most generic manner-,

$$\langle \underbrace{g_1, g_2, g_3, \dots, g_N}_{\text{optimisation parameters}}, \underbrace{\sigma_1, \sigma_2, \dots, \sigma_N}_{\text{step size parameters}}, \underbrace{C_{\sigma_1}, C_{\sigma_2}, C_{\sigma_3}, \dots, C_{\sigma_C}}_{\text{covariance parameters}} \rangle$$

where  $C$  is the number of elements in set  $C_{\sigma}$ .

### 2.2.2.2 Population in ES

As in the case for GA, the population is randomly initialized generally. Both steady state and generational population models can be used, but generational population models are more common.

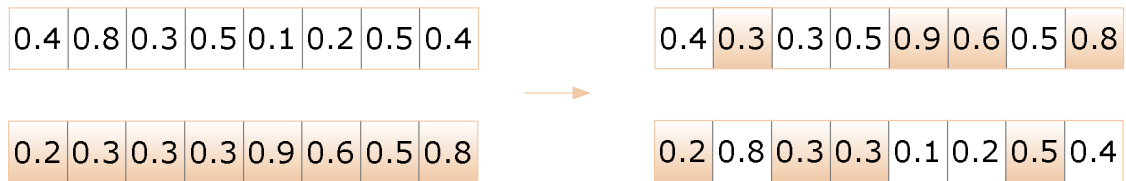
### 2.2.2.3 Parent selection in ES

There is not an actual selection routine for the parents as in GA. Because all the members are treated as parents, and whenever a parent is needed, it is drawn randomly (with uniform random distribution). That is to say, fitness scores (or rankings) are of no importance.

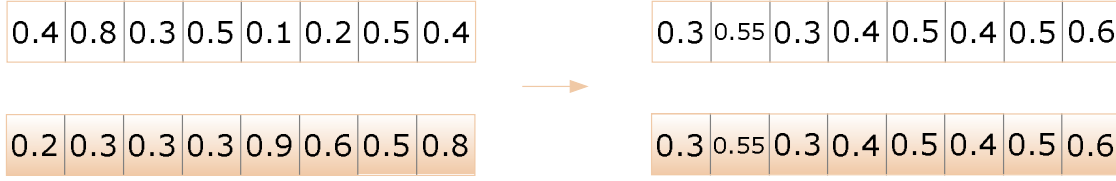
### 2.2.2.4 Variation operators in ES

### 2.2.2.5 Recombination operators

Basic recombination scheme for ES yields one offspring from two parents. In discrete recombination, the allele of a specific element in the offspring is directly copied from the same location of the randomly selected parent (**Figure 2.13**). In intermediate recombination, the values of each location are averaged from the parents (**Figure 2.14**). These basic recombination schemes utilizing two parents are called local recombination.



**Figure 2.13** : Discrete recombination in evolutionary strategies.



**Figure 2.14 :** Intermediate recombination in evolutionary strategies. (The scheme is the same with whole arithmetic recombination with  $\alpha = 0.5$  for real valued vectors in genetic algorithms.)

Extension to  $m$ -parents, where  $m \leq \mu$  (population size): the recombination operators describe above can be analogously applied for multi-parents. This is called global recombination for ES. Though this process does not match any real world process, in application it usually works better. There are many studies utilizing global recombination schemes as Beyer (1995), Schwefel and Rudolph (1995), Eiben and Back (1997), Back and Eiben (1999), Gruenz and Beyer (1999), Matsumura et al. (2001 and 2002).

In the literature, there are many reported recombination operator variants. One emerging idea is to use different recombination schemes for the optimization parameters and the strategy parameters. In fact, discrete recombination for the optimization parameters part is recommended to preserve diversity in the population. On the other hand, intermediate recombination has a more conservative tendency and provides more cautious adaptation of strategy parameters (Eiben and Smith, 2003).

A mutation operator adds a random number drawn from a normal (Gaussian) distribution to each allele. One-dimensional Gaussian distribution is (2.3),

$$N(x, \xi, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\xi)^2}{2\sigma^2}} \quad (2.3)$$

Then, the new element can be obtained by (2.4),

$$x_{i+1} = x_i + N(0, \sigma) \quad (2.4)$$

Remember that the step sizes will also undergo mutation then final equation can be re-written as (2.5),

$$x_{i+1} = x_i + N(0, \sigma_{i+1}) \quad (2.5)$$

where  $\sigma_{i+1}$  is the mutation step size after self mutation. That is to say, the mutation step size should be mutated first, and then it must be used to mutate optimization parameters. Therefore, the chromosome is effectively evaluated twice: In survivor selection, if this member is worth surviving in the next generation, then the mutation step sizes are somehow validated to yield fit members.

Selection of the strategy parameters and controlling their evolution through mutation operators generally depends on design experience. The categories for the mutation process can be divided into 3 basic branches,

- I. uncorrelated mutation with single step size,
- II. uncorrelated mutation with n step sizes and
- III. correlated mutations with n step sizes.

uncorrelated mutation with single step size :

$$\langle \underbrace{g_1, g_2, g_3, \dots, g_N}_{\text{optimisation parameters}}, \underbrace{\sigma}_{\text{step size parameter}} \rangle$$

uncorrelated mutation with  $n$  step sizes:

$$\langle \underbrace{g_1, g_2, g_3, \dots, g_N}_{\text{optimisation parameters}}, \underbrace{\sigma_1, \sigma_2, \dots, \sigma_N}_{\text{step size parameters}} \rangle$$

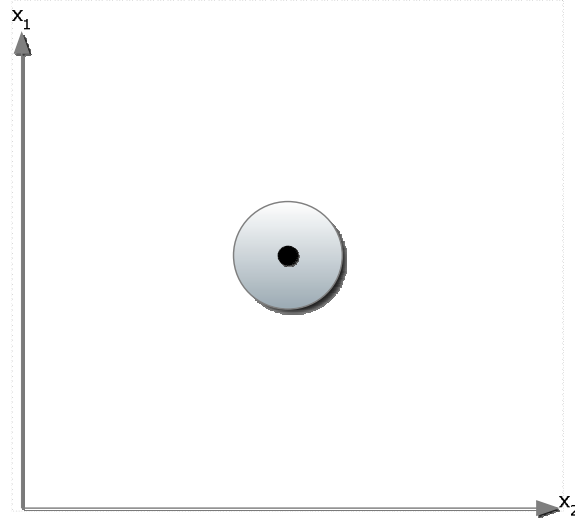
correlated mutations with  $n$  step sizes:

$$\langle \underbrace{g_1, g_2, g_3, \dots, g_N}_{\text{optimisation parameters}}, \underbrace{\sigma_1, \sigma_2, \dots, \sigma_N}_{\text{step size parameters}}, \underbrace{C_{\sigma_1}, C_{\sigma_2}, C_{\sigma_3}, \dots, C_{\sigma_c}}_{\text{covariance parameters}} \rangle$$

In single step size case,  $\sigma$  is mutated at each iteration by multiplying by a term  $e^\Gamma$  where  $\Gamma$  is a random number drawn from a normal distribution  $\tau$ .  $\tau$  is generally inversely proportional to the square root of the problem dimension and is analogous to the learning rate of neural networks. Then the mutation rules can be written in the correct order as in (2.6),

$$\begin{aligned} \sigma_{i+1} &= \sigma_i \cdot e^\Gamma \\ x_{i+1} &= x_i + N(0, \sigma_{i+1}) \end{aligned} \tag{2.6}$$

Note that by using Gaussian distribution with zero mean, mutation is not biased to either side and the probability of smaller modifications is more than that of larger ones. Single step size uncorrelated mutation is illustrated in **Figure 2.15**.



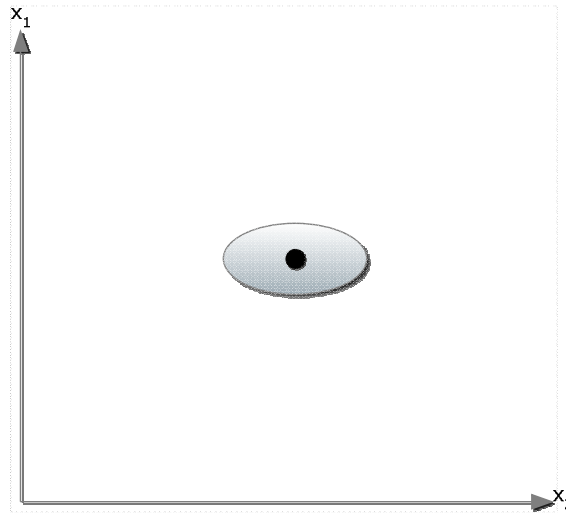
**Figure 2.15** : Representation of uncorrelated mutation with single step size for a two dimensional problem. The black dot represents a candidate solution (member) and the circle around it is the possible positions after mutation. Circle radius is related with  $\sigma$ . Note that the probability of moving in  $x_1$  axis is the same as moving in  $x_2$  axis.

If  $n$ -step sizes have been used, then the mutation effect on each dimension varies (**Figure 2.16**). In some problems that have different slopes in different dimensions, using multiple step sizes can be operational. The mutation process is as in (2.7),

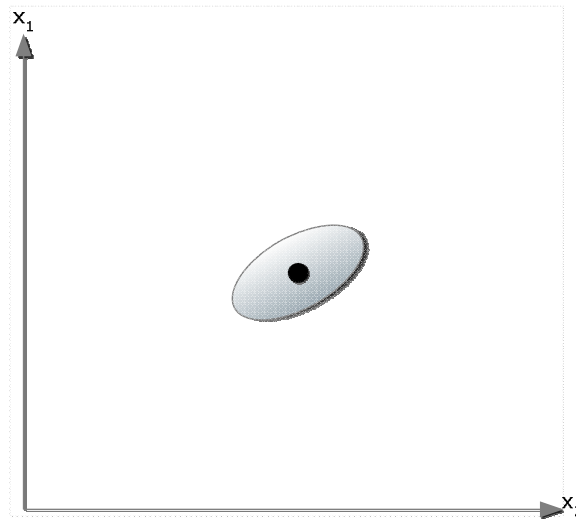
$$\begin{aligned}\sigma_{i+1}^{(j)} &= \sigma_i \cdot e^{\Gamma^{(j)}} \\ x_{i+1}^{(j)} &= x_i^{(j)} + N(0, \sigma_{i+1}^{(j)})\end{aligned}\tag{2.7}$$

Uncorrelated mutations are orthogonal in nature and so they are aligned with the axes. Correlation between dimensions defines rotation effect on the mutation (**Figure 2.17**). Correlated mutation is in (2.8),

$$\begin{aligned}\sigma_{i+1}^{(j)} &= \sigma_i \cdot e^{\Gamma^{(j)}} \\ \alpha_{i+1}^{(j)} &= \alpha_i^{(j)} + \lambda \cdot N(0, 1) \\ x_{i+1}^{(j)} &= x_i^{(j)} + N(0, \sigma_{i+1}^{(j)})\end{aligned}\tag{2.8}$$



**Figure 2.16 :** Representation of uncorrelated mutation with 2-step sizes for a two dimensional problem. The black dot represents a candidate solution (member) and the ellipse around is the possible positions after mutation. Minor axis of ellipse is aligned with  $x_1$ , major axis is aligned with  $x_2$ . These axes lengths are related with  $\sigma_1$  and  $\sigma_2$ . Note that the probability of moving in  $x_1$  axis is not the same as moving in  $x_2$  axis.



**Figure 2.17 :** Representation of correlated mutation with 2-step sizes for a two dimensional problem. The black dot represents a candidate solution (member) and the ellipse around is the possible positions after mutation. Neither of the axes is aligned with coordinates. Axes lengths are related with  $\sigma_1$  and  $\sigma_2$  and the rotation of the ellipse is related with  $\alpha$ . Note that the probability of moving in  $x_1$  axis is not the same as moving in  $x_2$  axis, but they are correlated.

Correlated mutations have the most parameters; however, effectively using these parameters is another matter of cost. Common approach is to start with uncorrelated

mutation with  $n$ -step sizes and then try moving to a simpler model if good results are obtained, or try moving to a correlation imposed one if results are not good enough.

In all mutation types, to avoid negligibly small standard deviations, a limiting value can be applied as in (2.9),

$$\sigma_{t+1} \leq \varepsilon_0 \Rightarrow \sigma_{t+1} = \varepsilon_0 \quad (2.9)$$

where  $\varepsilon_0$  is some user defined constant.

### 2.2.2.6 Survivor selection in ES

$\mu$  members for the next generation can be selected from the members of current generation plus the offspring population ( $(\mu + \lambda)$  selection) or if  $\lambda \geq \mu$ , they can be selected by only considering the offspring population ( $(\mu, \lambda)$  selection). Defining in terms of population concepts,  $(\mu + \lambda)$  selection causes a steady population model whereas  $(\mu, \lambda)$  selection causes a generational population model.

Generally  $(\mu, \lambda)$  selection is preferred in modern variants of the ES. In  $(\mu, \lambda)$  selection, dynamically changing fitness surfaces can be traced better, and it provides more efficient evolution of strategy parameters.  $\lambda$  is selected to be at around 5 to 10 times of  $\mu$ , therefore a great selection pressure is imposed in ES.

### 2.2.2.7 Self adaptation in ES

Self-adaptation of strategy parameters is the most critical aspect in ES. It has been firstly proposed as an ES issue and has been investigated for certain effects in ES algorithms. Now, its usage is widespread in EA society. Its benefits have been shown not only for real valued representations but also for binary and integer representations (Back, 2000). Theoretical (Beyer, 2001) and experimental results on self-adaptation clearly states that the standard deviation of the random number added at each iteration must decrease. By intuition, at the very first steps of the search, the algorithm is not intensified (focused on a specific point) but can even check the furthest places in the search space with higher probability. Then, as the iterations elapse, the search is focused on the specific regions of suspect by decreasing the probability of checking further points.



Academic studies up to now define the necessary conditions for self-adaptation as the following (Eiben and Smith, 2003),

1.  $\mu > 1$  so that different strategies are present
2. Generation of an offspring surplus:  $\lambda > \mu$
3. A not too strong selective pressure (heuristic:  $\lambda / \mu \approx 7$ )
4.  $(\mu, \lambda)$ -selection (to guarantee extinction of misadapted individuals)
5. Recombination also on strategy parameters.

### 2.2.3 Big Bang – Big Crunch optimization algorithm

Big Bang-Big Crunch (BB-BC) optimization algorithm is a global optimization method inspired by two of the main theories on the formation of the universe, namely Big Bang and Big Crunch theories. It was proposed by Erol and Eksin, (2006).

BB-BC optimization method is a population based evolutionary algorithm. By the very first big bang, the individuals of the population are dispersed throughout to the search space in a random uniform manner. That is to say, *big bang phase* of the first iteration is randomly initializing the population members. This is done by adjusting the random number generators to cover only the search space of interest. Then in the following big crunch phase, a representative point (or representative member) is generated by using information from all of the members of the population. Representative point of the iteration is named as the centre of mass. Big crunch phase can be represented as multi input single output function and the formulation of the crunching process for a minimization problem can be simply given as in (2.10),

$$\vec{x}^c = \frac{\sum_{i=1}^N \frac{1}{f^i} \vec{x}^i}{\sum_{i=1}^N \frac{1}{f^i}} \quad (2.10)$$

where,  $x^c$  is the centre of mass (representative point),  $\vec{x}^i$  is the position vector for the  $i^{th}$  individual,  $f^i$  stands for the fitness value of the  $i^{th}$  individual and  $N$  is the population size. Therefore, crunching operation is equivalent to taking weighted

average of the individual positions with respect to inverse of the fitness scores assigned.

Big Bang and Big Crunch phases are performed at each iteration of the search. In the second and the following iterations, new generation of population is created by using the weighted sum obtained in the previous big crunch phase. New members are calculated around the centre of mass by adding or subtracting a random number drawn from a Gaussian distribution whose value decreases as the iterations elapse. More precisely, the probability of having large random numbers is decreased by modifying the standard deviation of the Gaussian distribution and as a result, the probability of reaching the further corners of the search space is much more in comparison to that at the final iterations. The size of this added (or subtracted) value is analogous to the explosion strength of a physical explosion process. This dynamic behavior provides more diversification when there is little knowledge in the first few iterations and then causes intensified search around the suspected global minimum at the final iterations. Note that, even in the final iteration of the search, there is a certain (and probably very limited) probability for reaching far corners of the search space.

Each member of the new generation (=population of the next iteration) can be derived by (2.11),

$$x^{new} = x^c + lr/k \quad (2.11)$$

where  $l$  is the upper limit of the parameter,  $r$  is a normal random number and  $k$  is the iteration step. Then, the new point  $x^{new}$  is upper and lower bounded to fit into the search space.

As is the case for all evolutionary iterative algorithms, the algorithm runs until a predefined stopping criterion has been met. Among the commonly used stopping criteria are,

- I. maximum number of iterations,
- II. maximum number of fitness evaluations,
- III. maximum allowed run time,
- IV. minimum convergence goal for the fitness values,

V. minimum convergence goal for the population member positions.

The stopping criteria can be selected problem specifically. Basic BB-BC algorithm can be utilized to stop once a predetermined number of iterations elapsed.

The algorithm steps can be summarized as follows,

STEP 1: Form the initial population of  $N$  members distributed uniformly in the search space.

STEP 2: Assign a fitness value for all the members.

STEP 3: Calculate the representative point by using **(2.10)**.

STEP 4: Calculate the new members of the next generation by adding or subtracting a random number drawn from a Gaussian distribution whose value decreases as the iterations elapse.

STEP 5: Check for the stopping criterion: if it has been met, stop; else go back to step 2.

In the originating paper for the BB-BC optimization algorithm (Erol and Eksin, 2006), simulation results on benchmark test functions are reported. The tests are carried for same iteration number, same fitness evaluation number and same run time. The algorithm had been proven to possess the quick convergence capability even in the long, narrow parabolic shaped flat valleys or in the existence of several local minima. Though it is a new algorithm, it has been applied to many areas including target motion analysis problem (Genç & Hocaoglu, 2008), fuzzy model inversion (Kumbasar et al, 2008; Kumbasar et al, 2008), design of space trusses (Camp, 2007), size reduction of space trusses (Kaveh and Talatahari), airport gate assignment problem (Genç et al, 2009), non-linear controller design (Dogan & Istefanopulos, 2007) and genetic programming classifier design (Akyol et al, 2007).

#### **2.2.4 Relations and differences with the previously reported literature**

There are tremendous amount of work carried out in the evolutionary computing society. Many components of the different algorithms have certain relationships with the others. This section is dedicated to report similar routines with the BB-BC algorithm or with a certain part of the BB-BC algorithm.

#### **2.2.4.1 Similar genetic algorithm approaches**

Creep mutation for integer representations and non-uniform mutation for floating point representations: In both mutation routines, a randomly drawn number from a specific distribution (mostly Gaussian distribution) is added or subtracted from the genes. These mutations require different parameters for controlling the distribution and hence the size of the steps that mutation takes in the search space. This aspect of the mutation operators has great similarities with the banging phase of the BB-BC where randomly drawn numbers are added or subtracted from the center of mass. Moreover, the sizes of the perturbations (size of the added or subtracted numbers, explosion strength in BB-BC terminology) are controlled with a single parameter in original BB-BC that decreases as the iterations elapse. Similar approaches are reported in the literature for the creep mutation and the non-uniform mutation (Zhao and Gao, 2004; Clemente et al, 2003, Neubauer, 1997).

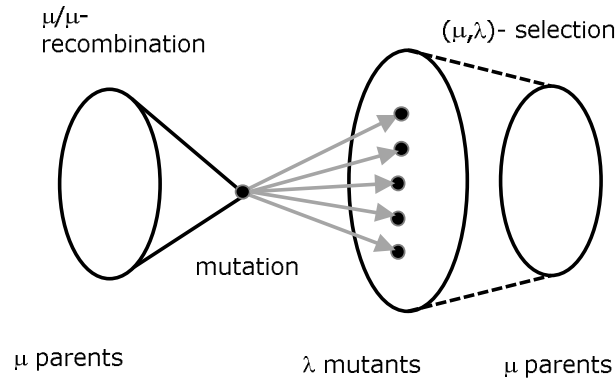
Multi-parent arithmetic crossover (MAC): It is a multiparent arithmetic crossover for real valued (floating point) representations proposed by Mendoza et al. (2001). MAC is the generalized form of arithmetic recombination designed for  $P$ -parents. The crunching phase of the BB-BC algorithm is a specific version for this crossover working with  $N$ -parents (where  $N$  is the number of individuals in the population) and yielding only single offspring (that is named as the centre of mass in BB-BC optimization method).

#### **2.2.4.2 Similar evolutionary strategies approaches**

Uncorrelated mutation with single step size: Concept of self-evolution of the strategy parameters of ES is actually not implemented in BB-BC algorithm. Yet, there is a single strategy parameter controlling the magnitude of the random number ( $\sim$ step size) generated. This number is used for every dimensions of the search space. In that manner, there is single step size in BB-BC.

Global recombination for ES: Intermediate recombination scheme can be expanded to more than two parents or more than two donors. Specifically, Beyer (1995) proposed to use all the members of the population to generate a centre of mass. This centre of mass is then used to produce  $\lambda$  offspring by mutating the individual with mutation vector  $\mathbf{Z}$  of length  $\lambda$ . This ES variant is named as  $(\mu/p, \lambda)$ , meaning that,  $\mu$  parents (whole population) generate  $\lambda$  offspring through recombination and mutation

at each generations.  $\lambda \geq \mu$ ; so that best  $\mu$  offspring is selected deterministically for the next generation.  $p$  is the number of donors (number of parents to form one new offspring) and in Beyer's formulation  $p = \mu$  (**Figure 2.18**). Beyer proposed to use mutation vector  $\mathbf{Z}$  as generated by independent and identically distributed normal random numbers with zero mean and standard deviation,  $\sigma_i$ , for each component (Beyer, 1995). BB-BC is a reformulation of this multi parent ES variant that can be symbolized as  $(\mu/\mu, \mu)$  and single  $\sigma$ .



**Figure 2.18 :**  $(\mu/\mu, \lambda)$  recombination in general scheme.

Beyer notes that this centre of mass operation enforces an extreme reduction of diversity, which could be expected to have a negative effect on convergence, reliability or the self-adaptation capability of the algorithm. Many other researchers utilizing global intermediate recombination have obtained similar facts as a result of their research work (Back and Eiben, (1999), Gruenz and Beyer (1999), Matsumura et al. (2001 and 2002)).

BB-BC optimization originates from the Big Bang Theory; but ends up with nearly the same algorithm routine with global intermediate recombination in ES. The mutation routine for BB-BS is nonlinearly decreasing explosion strength that is correlated with the standard deviation (or single mutation step size).

Crunching phase of BB-BC can select the fittest member of the population as the representative point. This approach is another ES variant.

The performance comparison between the two BB-BC crunching phase variants (selecting the fittest or weighted averaging by (2.10)) in unimodal / multimodal problems reveals that there are no considerable accuracy difference between the two, but fittest selection is faster (**Table 2.1**).

**Table 2.1** : Average costs for multimodal test functions,  $n = 20$ , 1000 evaluations, search space:  $[-10, 10]$  for both parameters. (Average of 1000 runs).

| FUNCTION   | Explosion<br>Centre: Fittest<br>Member | Explosion<br>Centre:<br>Weighted<br>Average | Explosion<br>Centre: Average |
|------------|--|---|------------------------------|
| Ackley     | 0.61925                                | 0.62079                                     | 0.91577                      |
| Griewank   | 0.01483                                | 0.01278                                     | 0.00889                      |
| Rastrigin  | 1.32720                                | 1.29570                                     | 1.70310                      |
| Rosenbrock | 0.37051                                | 0.33529                                     | 0.31980                      |
| Schwefel   | 0.00194                                | 0.00188                                     | 0.00335                      |

Beyer, in 2001 (Beyer, 2001) also proposed the weighted average recombination for global intermediate recombination. In the most general manner, the recombination output at each generation (=centre of mass) is the weighted average of samples in the parent population, **(2.12)**,

$$\vec{c} = \sum_{j=1}^{\mu} w_j x_j \quad (2.12)$$

where  $\vec{c}$  is the  $d$ -dimensional centre of mass vector;  $w_j$ 's are weighting coefficients such that all  $w_j$ 's sum up to 1. Based on such intermediate recombination scheme different ways of determining the weights have been proposed (Salomon, 1998; Arnold, 2004; Arnold and MacDonald, 2006; Hansen and Ostermeier, 2001). For example, if all  $w_j$ 's are selected to be  $1 / \mu$ , then the centre is simple the average of all members; if they are fitness related **(2.13)**,

$$w_j = f_j / \sum_{j=1}^{\mu} f_j \quad (2.13)$$

then the centre is the mean of all individuals weighted with respect to fitness scores as in BB-BC. Using weighted mean as recombination centre is used in one of the most commonly used variant of ES, namely CMA-ES (Covariance Matrix Adaptation Evolution Strategy) algorithm introduced by Hansen (Hansen and Ostermeier, 2001; Hansen et al, 2003).

### **3. BIG BANG – BIG CRUNCH OPTIMIZATION WITH LOCAL DIRECTIONAL MOVES**

#### **3.1 Local Search in Evolutionary Computation**

Memetic algorithms (MA) represent one of the recent growing areas of research in evolutionary computation. The term Memetic Algorithms has first appeared in the computing literature in 1989 (Moscato, 1989). The rationale behind MAs is to provide an effective and efficient global optimization method by compensating for deficiency of evolutionary algorithms (EA) in local exploitation and inadequacy of local search (LS) in global exploration (Noman and Iba, 2008). The term MA is now widely used for any population-based approach with separate local improvement procedures.

Real coded memetic algorithms are classified into two main classes depending on the type of LS employed (Lozano et al, 2004):

- 1) Local improvement process (LIP) oriented LS (LLS): This category refines the solutions of each generation by applying efficient LIPs, like gradient descent. LIPs can be applied to every member of the population or with some specific probability and with various replacement strategies.
- 2) Crossover-based LS (XLS): This group employs crossover operators for local refinement. A crossover operator is a recombination operator that produces offspring around the parents. For this reason, it may be considered as a move operator in an LS strategy (Lozano et al, 2004).

Adaptation of parameters has become a very promising research field in MAs. Ong and Keane (2004) proposed meta-Lamarckian learning in MAs that adaptively chooses among multiple memes during a MA search. They proposed two adaptive strategies in their work and empirical studies showed their superiority over other traditional MAs. A taxonomy and comparative study on adaptive choice of memes in MAs is presented in Ong et al. (2006). In order to balance between local and genetic search, Bambha et al. (2004) proposed simulated heating that systematically

integrates parameterized LS (both statically and dynamically) into EAs. Ahn et al. (2010) also applied adaptive local search routine to multi-objective evolutionary optimization problems. The common aspect for all the memetic methods proposed so far is that they needed mechanisms that have to,

- I. decide the step length (and adaptation of step length) of the local search,
- II. draw a balance between exploration and exploitation; that is, local search and global search.

A comprehensive review on hybrid genetic algorithms can be found in El-Mihoub et al. (2004).

In this chapter, a new memetic algorithm is introduced in which a local search is imposed between the phases of the BB – BC optimization method and the crunching phase is improved by the addition of Nelder-Mead method to calculate fittest point of the iteration. The local search algorithm generates a direction vector by using the current fittest point and the previous fittest points of the generations and checks for improvement in this direction. If an improvement is achieved, the new centre is forced to switch to that point. That is to say, the centre point of the explosion of next big bang phase is changed. Note that, by using the distance between these consecutive representative points, the step size of the local search is set and adjusted accordingly. Local search enhances the exploitation or intensification capability of the algorithm; and thus, the proposed hybridization operation produces much more accurate results than the original BB – BC algorithm. In fact, it also provides promising results when compared to the state-of-the-art optimization methods. Moreover, the newly proposed algorithm is shown to be much more effective in terms of complexity.

The rest of the chapter is divided into four subsections. Effect of Nelder - Mead crunching and local directional moves are given first; then the newly proposed hybrid method is given as a complete algorithm. The simulation results on various test functions are presented to illustrate the effectiveness of the new hybrid algorithm. Possible further developments and conclusions are finally elaborated and discussed in the last subsection.



## 3.2 Inspection of the Effect of Nelder-Mead Crunching

### 3.2.1 Big bang-big crunch algorithm with Nelder-Mead crunching

For unimodal problems, the crunching phase of the original BB-BC optimization algorithm can be improved to end up with a better center of mass. The original BB-BC method either uses a weighted sum for the population members or it simply takes the fittest member as the representative point. Instead, a more complex local search routine, namely Nelder and Mead method can be used (Genç, 2010; Genç et al, 2010a). Nelder and Mead method (Nelder and Mead, 1965) is a simplex method for finding a local minimum (maximum) of a function of several variables. For two variables, this simplex becomes a triangle, and the method is a pattern search that compares function values at the three vertices of a triangle (Mathews and Fink, 2004).

In the proposed method, crunching is performed as the result of Nelder Mead Method. The worst vertex, where objective value is largest, is rejected and replaced with a new vertex. A new triangle is formed and the search is continued. The process generates a sequence of triangles (which are not necessarily regular), for which the function values at the vertices get smaller. The size of the triangles is reduced and the coordinates of the minimum point are found (Mathews and Fink, 2004). That is to say, Nelder and Mead method is used as a centre of mass operator of the original BB-BC algorithm.

At each iteration, after big bang phase, three vertices are chosen to form the simplex: the fittest member ( $B$ ), the second fittest member ( $G$ ) and the worst member ( $W$ ). The hard constraint on algorithm construction is that the population size must be greater than or equal to three (and greater than or equal to  $n + 1$  for  $n$ -dimensional problems). Then the Nelder and Mead algorithm steps for a two-dimensional minimization problem can be given as the following (Mathews and Fink, 2004) and the basic moves of the algorithm; reflection, contraction, expansion and shrinking are illustrated in the **Figure 3.1-Figure 3.4**.

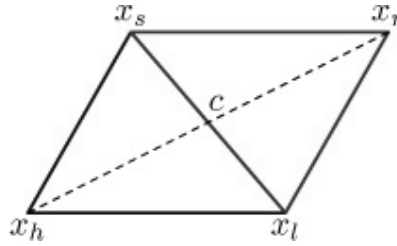
STEP 1: Construct the initial triangle with vertices  $X_l$ ,  $X_s$  and  $X_h$ .

STEP 2: Calculate centroid  $C$  for reflection

$$C = (X_l + X_s) / 2 \quad (3.1)$$

STEP 3: Calculate reflection point  $X_r$  for getting away from  $X_h$  and compare  $f(X_r)$  and  $f(X_s)$

$$X_r = C + (C - X_h) \quad (3.2)$$

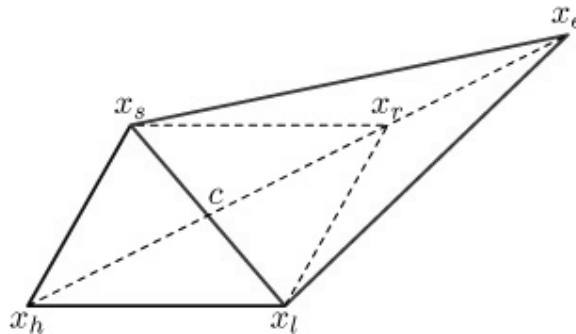


**Figure 3.1 :** Reflection to point  $X_r$ .

- a. If  $f(X_l) < f(X_r) < f(X_s)$ , then replace  $X_h$  with  $X_r$  (reflection move)
- b. If  $f(X_r) \leq f(X_l) < f(X_s)$ , then compute expansion point  $X_e$ , (3.3)

$$X_e = X_r + (X_r - X_h) \quad (3.3)$$

If  $f(X_e) < f(X_r)$ , replace  $X_h$  with  $X_e$  (expansion move) otherwise perform reflection move.

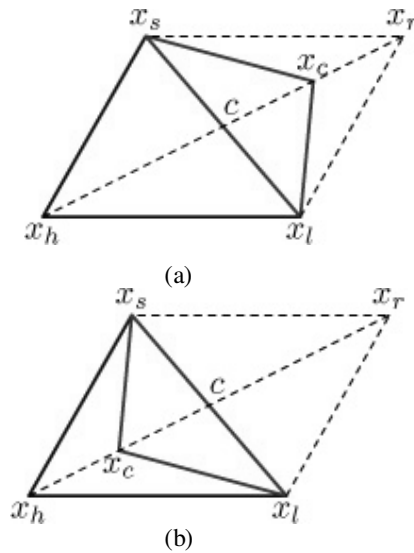


**Figure 3.2 :** Extension to point  $X_e$  from  $X_r$ .

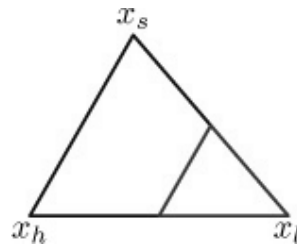
- c. If  $f(X_h) > f(X_r) \geq f(X_s)$ , replace  $X_h$  with  $X_r$  and compute contraction point  $X_c$ , (3.4),

$$X_c = (X_h + C) / 2 \quad (3.4)$$

- i. If  $f(X_c) < f(X_h)$ , replace  $X_h$  with  $X_c$
- ii. If  $f(X_c) \geq f(X_h)$ , compute shrinking points and replace these points with  $X_h, X_s$



**Figure 3.3 :** Contraction points  $X_c$ . (a) Outside contraction, (b) Inside contraction.



**Figure 3.4 :** Shrinking towards  $X_l$ . Point  $X_s$  and point  $X_h$  comes closer to  $X_l$ .

STEP 4: Check the termination criterion (3.5)

$$|X_l - X_h| < tolerance \quad (3.5)$$

If the termination criterion is satisfied, then the local search step terminates resulting point  $X_l$  as the centre of mass for the current iteration. If not, go back to step 2.

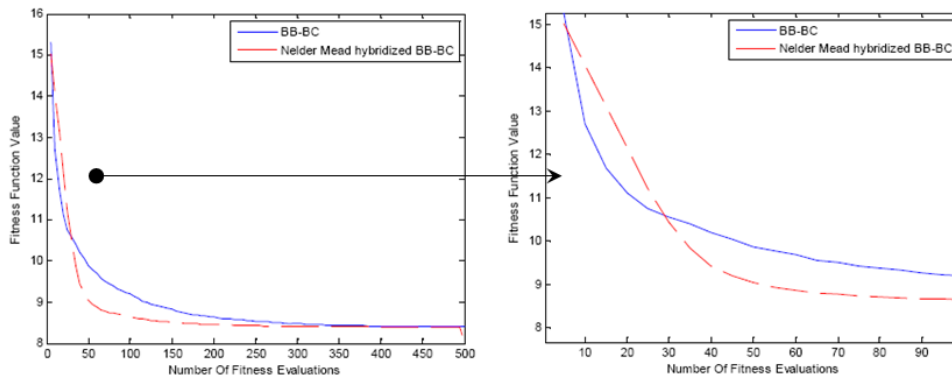
The tolerance value used in step 4 of Nelder Mead algorithm flow is dynamically updated with respect to iteration count. As the iteration number for the overall algorithm increases, tolerance value for the corresponding Nelder-Mead run is decreased. Therefore, initial iterations have less intensive crunching search phases. Parameters affecting the explosion strength also control of the tolerance; that is no new parameters are introduced over the original BB-BC method.

Test for convergence (or termination) can be carried on numerous ways. Tests based on

- I. the standard deviations of the three points,
- II. the closeness of fitness values,
- III. the improvement gained,
- IV. the limitation of fitness evaluations,
- V. the limitation of iterations or
- VI. any combination of all these

can be used. The aim of this hybridization scheme is to fasten the search by checking some local points in the crunching phase of the main global search algorithm so as to maximize improvement probability. The neighboring points check procedure should be carried in a guided manner (provided by using Nelder and Mead optimizer) and for limited points.

NM crunching is a new approach in order to improve exploitation capability near a local minimum. If NM crunching is used at the initial iterations of the search then excessive function evaluations has to be performed since there is not enough knowledge on the function topology or coverage. Then, it is better not use NM crunching at the early iterations of the search algorithm; instead, this crunching method should be switched when the search has evolved and ripen. The effect of NM crunching after the initial iterations is illustrated in the convergence graph in **Figure 3.5** for a multi-modal function.



**Figure 3.5 :** Convergence graphs for BB-BC algorithms using best point as the centre of mass (solid line) and the NM method (dashed line).

### 3.2.2 Simulation results for Nelder Mead crunching

The results of the addition of Nelder-Mead crunching are compared with results of the original BB-BC optimization algorithm on the Ackley, Rastrigin, Rosenbrock test functions. The stopping criterion is defined as the maximum number of fitness

evaluations for both algorithms. If one would have chosen the stopping criteria as number of iterations then the hybrid algorithm would have been advantageous compared to the pure BB-BC algorithm since the new hybrid algorithm searches for extra points around fittest individual in crunching phase; and therefore, the comparison would not have been fair. Utilizing the local search step instead of using weighted average method (or directly selecting the fittest member as the representative point) makes the algorithm slower but in most of the practical problems, main process time is spent in the cost function evaluation. Therefore, fitness evaluation time makes the other steps negligible. (In basic benchmark functions used, that is not the case though.) The time spent for both original and hybrid BB-BC Algorithms are reported in the **Table 3.1**-Table 3.4.

The results logged in this chapter are obtained from 10000 random run for each test. This number is more than enough for reliable statistical analyses. **Table 3.1-Table 3.3** report the results with respect to the objective functions. The simulations are carried for different population sizes, different number of evaluations before termination and for different sizes of total search spaces. Here, the reported results are for 20 individuals allowed for 1000 objective evaluations. The search space is [-10, 10] for both parameters. The tremendous improvement can be easily observed from both the average and median values for the total runs. Standard deviations of the results are also smaller for the newly proposed method, which makes it more consistent.

**Table 3.1** : Ackley test function results.

| ACKLEY                      | BB-BC | BB-BC with<br>Nelder-Mead |
|-----------------------------|-------|---------------------------|
| Average Cost                | 0.625 | 0.022                     |
| Median Cost                 | 0.529 | 0.023                     |
| Std. Dev. Of Cost           | 0.430 | 0.005                     |
| Average Time<br>Elapsed (s) | 0.004 | 0.081                     |

**Table 3.2 : Rastrigin test function results.**

| RASTRIGIN                | BB-BC | BB-BC with Nelder-Mead |
|--------------------------|-------|------------------------|
| Average Cost             | 1.314 | 0.415                  |
| Median Cost              | 1.263 | 0.010                  |
| Std. Dev. Of Cost        | 0.711 | 0.561                  |
| Average Time Elapsed (s) | 0.003 | 0.069                  |

**Table 3.3 : Rosenbrock test function results.**

| ROSENBROCK               | BB-BC | BB-BC with Nelder-Mead |
|--------------------------|-------|------------------------|
| Average Cost             | 0.386 | 0.004                  |
| Median Cost              | 0.135 | 0.003                  |
| Std. Dev. Of Cost        | 0.918 | 0.002                  |
| Average Time Elapsed (s) | 0.002 | 0.067                  |

In **Table 3.1-Table 3.3**, there is great amount of difference on average time elapsed values. For this reason, another test is designed: the algorithm is terminated when the fittest member of the iteration comes to 0.1 vicinity of the global minimum (Global minimum value is known at the beginning for these benchmark functions). The results of the tests are given for only the Ackley function in **Table 3.4**. The other functions behave the same. In **Table 3.4**, the original algorithm needs %50 more time to process in order to achieve similar accuracy with the newly proposed hybrid algorithm. The difference between the evaluation numbers is also notable.

**Table 3.4 :** Ackley test function results for the second termination criterion (average and median errors < 0.1).

| ACKLEY                            | BB-BC  | BB-BC with Nelder-Mead |
|-----------------------------------|--------|------------------------|
| Average Cost                      | 0.062  | 0.063                  |
| Median Cost                       | 0.065  | 0.063                  |
| # of av. fitness evaluations      | 2663.7 | 120.1                  |
| # of iterations carried (average) | 133.18 | 2.878                  |
| Average Time Elapsed (s)          | 0.012  | 0.008                  |

### 3.3 Inspection of the Effect of Improvement Vectors

#### 3.3.1 Big Bang-Big Crunch algorithm with improvement vectors

Bang Big-Big Crunch optimization method can be further improved by using local search routines in conjunction with the original phases of the algorithm. Utilizing local search in between the algorithm iterations is a simple yet effective way of achieving this (Genç et al, 2010b). As local search module, a direction is generated by using the current and the previous representative members of the population and this search line is further investigated in the aim of obtaining a better representative point. If the search terminates without improving the current best solution at hand, the algorithm run simply continues with the next iteration of the BB-BC algorithm; else, the obtained new point on the search space replaces the representative point. The global search part of the algorithm, that is the BB-BC algorithm reviewed in the previous chapter, has been preserved and applied with no modification within itself or its parameters: Between the iterations of BB-BC, local search step is injected.

The steps of the algorithm are,

STEP 1: Form the initial population of  $N$  members distributed uniformly in the search space.

STEP 2: Perform the crunching phase of the BB-BC algorithm. This point becomes the first best point found in the iteration. Store this point.

STEP 3: Perform once more the consecutive banging and then crunching phases of the BB-BC algorithm.

STEP 4: If the best point obtained in step (3) is better than the last stored point then this means an improvement then store that point. Next, generate a direction vector using one or two previous best candidate solution points so far attained, make exploratory moves in that direction, and assign a new virtual centre of mass on that direction if a better point has been obtained than the previous fittest point. If the best or the fittest point remains the same after the local search phase then go straight to step (5).

STEP 5: Check the stopping criteria. If it is met stop; else go back to step (3).

The proposed idea with this hybridization scheme is to fasten the search for global minimum. Once a search direction is obtained, a few points on this line are checked to look for any better points. The search is not intensive, so finding the exact local minimum is not the ultimate goal. Instead, a better starting point for the next iteration (or a better representative point for the current iteration) is tried to be obtained. Thus, the next explosion centre of the bang phase is not guaranteed to be a local minimum. Moreover, the big bang phase of the BB-BC algorithm is still global in nature and these two factors avoid search stagnation. Note that, local search is performed not randomly, but along an improving line by using commonly accepted contraction and expansion moves or dichotomous search. Otherwise, checking random neighbors or complete set of neighbors can cause unacceptable processing time or even search stagnation. **Figure 3.6** gives the flowchart for the algorithm in a generic manner.

Three different approaches for the local search part of the algorithm are reported in the following subchapters.

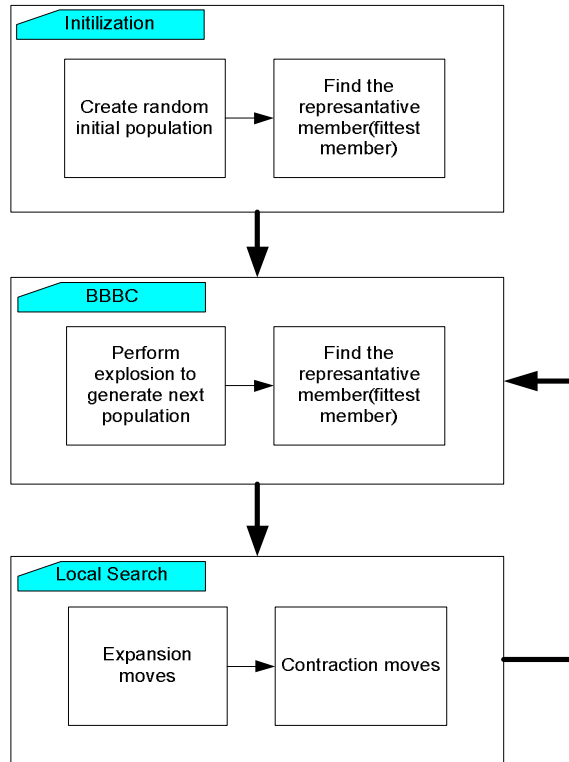
### 3.3.1.1 Vector formation with single step regression

In single step regression, the direction vector (also named as improvement vector) is the difference vector of the fittest point of current iteration and the previous best (fittest, representative) points stored after the last two consecutive crunching phases of the BB-BC algorithm, (3.6),

$$IV_1 = P(n) - P(n-1) \quad (3.6)$$

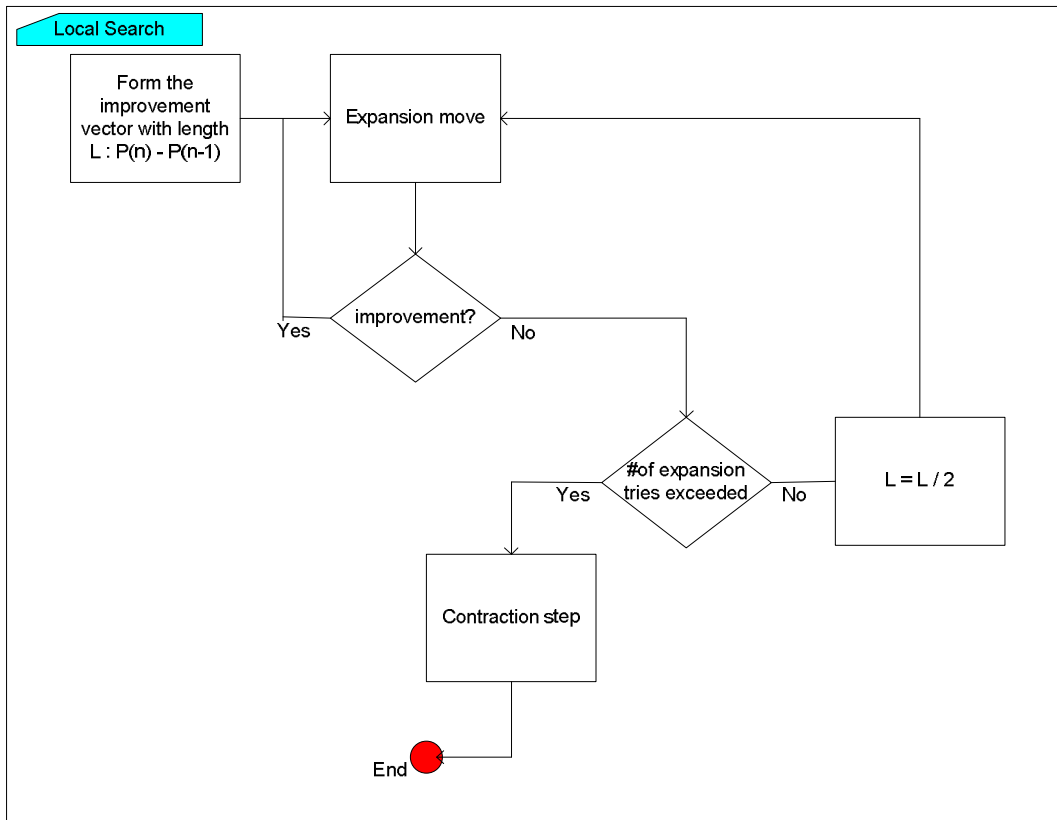


where  $IV_l$  stands for the improvement vector of single step regression BB-BC,  $P(n)$  is the current best or fittest point and  $P(n-1)$  is the last stored best or fittest point.

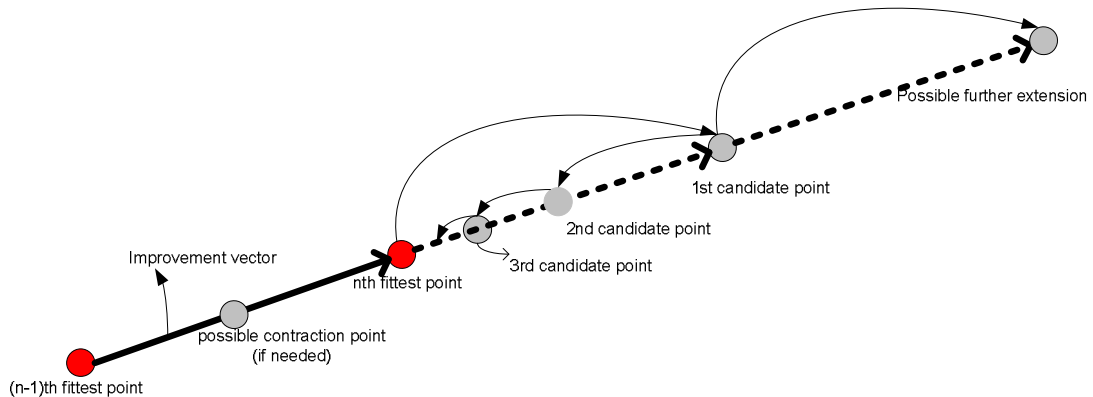


**Figure 3.6 :** Generic algorithm flowchart.

In this version of local search methodology, the memory usage is just for single step; and therefore, there is no information usage from the representative points belonging to the previous iterations. In the search methodology, the magnitude of the direction vector is halved after each unsuccessful expansion step. User should determine the number of halving operations. If all the expansion trials turn out to be a failure, only one contraction operation is allowed. None of these predetermined parameters within these local move operations are hard constraints for algorithm and they can be relaxed when needed with respect to the problem geometry. The flowchart of the local search part is given in **Figure 3.7** and the search steps on the direction line are illustrated in **Figure 3.8**.



**Figure 3.7 :** Local search phase for single step regression in BB-BC algorithm.



**Figure 3.8 :** Illustration of direction vector formation for local improvement with single step regression in BB-BC Algorithm.

### 3.3.1.2 Vector formation with double step regression

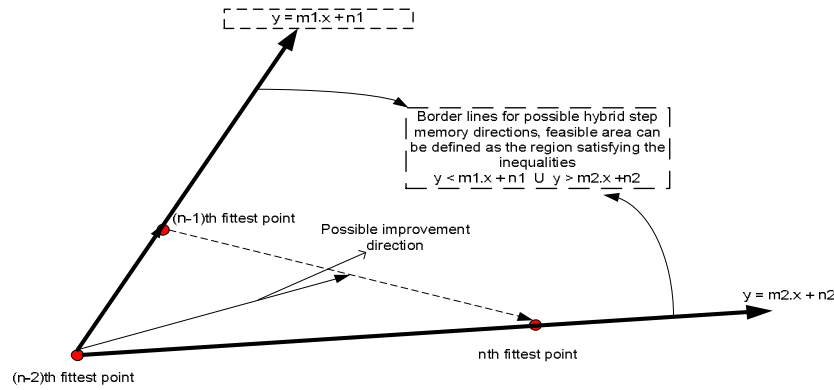
In double step regression, the direction vector is the weighted mean of  $IV_1$  and  $IV_2$  where  $IV_2$  is defined similar to  $IV_1$ , (3.7),

$$IV_2 = P(n) - P(n-2) \quad (3.7)$$

$$IV_h = \alpha IV_1 + (1 - \alpha) IV_2$$

where  $\alpha$  is a number in the interval  $[0, 1]$ . Note that if  $\alpha = 1$ , double step regression procedure reduces to single step regression. There is information usage from both the  $(n-1)^{th}$  and  $(n-2)^{th}$  representative points; thus, this provides to form non-regular simplex for local minimum search.

The bounds for the search direction are illustrated in **Figure 3.9**. In the figure, a possible direction vector is given for the case  $\alpha = 0.5$ .



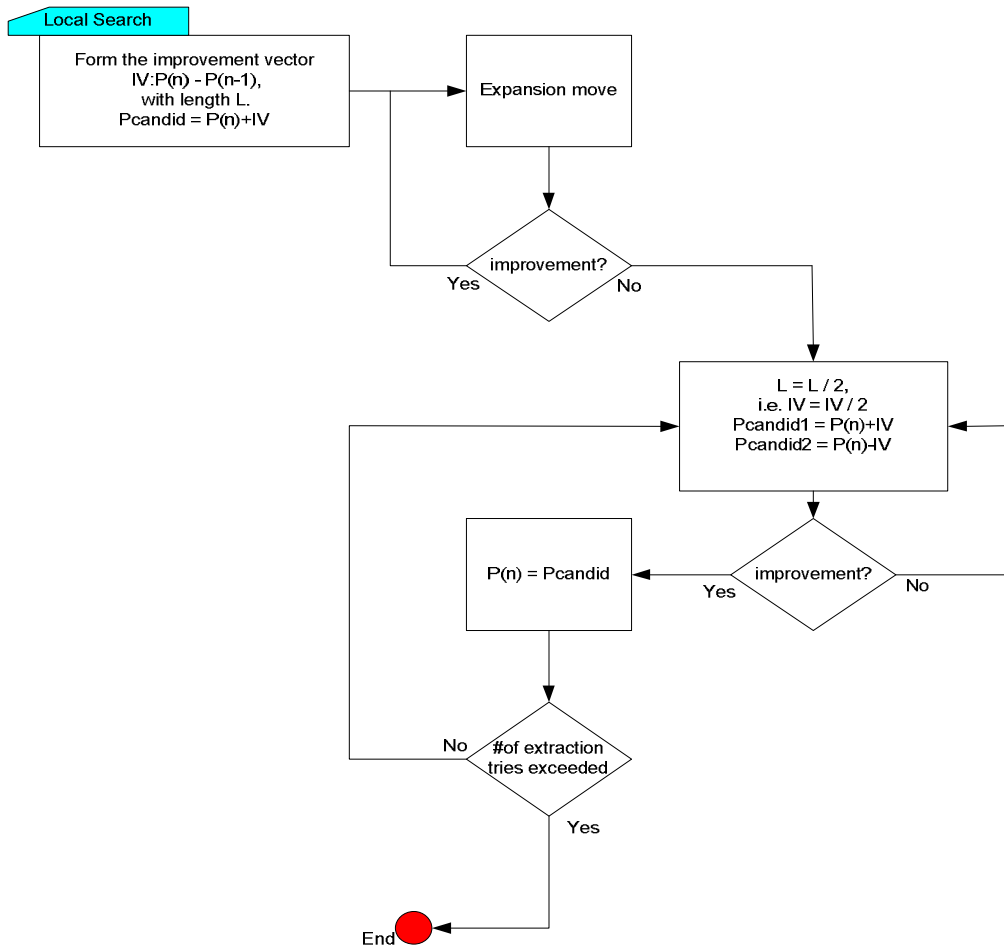
**Figure 3.9 :** Illustration of direction vector formation for local improvement with double step regression in BB-BC Algorithm.

### 3.3.1.3 Dichotomous search on local direction vector

Instead of checking extraction and contraction points, dichotomous search technique can be utilized on the generated search line. The flowchart for dichotomous search on local direction vector for one step regression in BB-BC Algorithm is illustrated in **Figure 3.10**.

**Figure 3.11** serves as an illustration example of the representative point evolutions in applying both the original BB-BC Algorithm and the hybrid BB-BC Algorithm. In this example, the original BB-BC Algorithm and the hybrid BB-BC Algorithm has been run on the same objective function (Rosenbrock objective function: minimum at  $(x=1, y=1)$ , minimum cost = 0) with same parameters and same random number generator seeds and it starts from the same point  $(x_1= 2.4721, y_1=6.3589)$ . In the

following iterations the fittest point moved to another location where is shown as  $p$  in **Figure 3.11**. While the original BB-BC Algorithm performs the next explosion centering this point, the proposed hybrid algorithm replaces point  $p$  with  $p'$ . The same procedure follows for the whole run and the resulting trajectories are given in **Figure 3.11**. The hybrid BB-BC Algorithm clearly ends up in a closer point to the global minimum at (1, 1) with a smoother trajectory. Note also that this simple example is given for the direction vector formation with single step regression case.

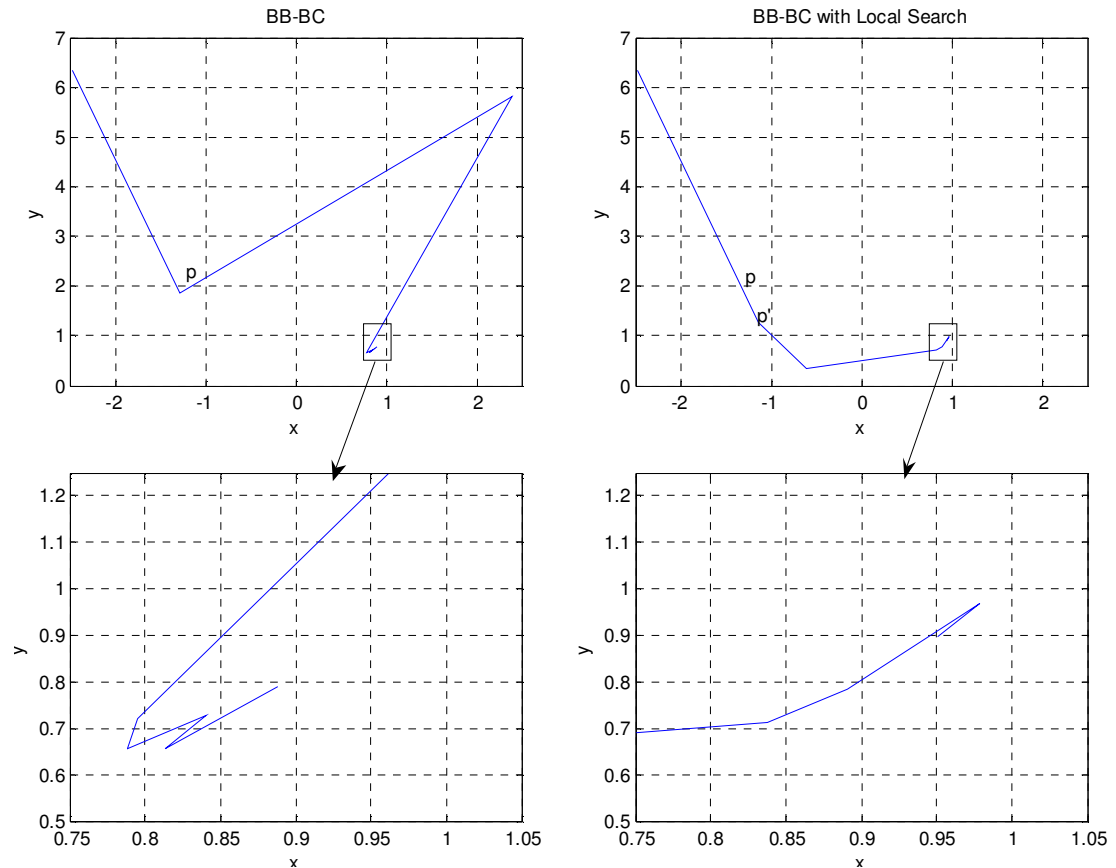


**Figure 3.10** : Flowchart for the dichotomous local search algorithm.

### 3.3.2 Simulation results for improvement vector generation

The results of the hybrid method are compared with results of the original BB-BC optimization algorithm on the objective test functions; namely, rosenbrock, rastrigin, ackley, sphere, step and ellipsoid functions. All these test functions are chosen to be same with the original paper presenting the BB-BC Algorithm (Erol & Eksin, 2006). The stopping criterion is defined as the maximum number of fitness evaluations for

both algorithms. The time spent for both original and hybrid BB-BC Algorithms are almost the same; and therefore, this criteria is not taken into consideration for comparison purposes.



**Figure 3.11 :** The hybrid BB-BC Algorithm versus the original BB-BC Algorithm - Upper left hand side: movement of the original BB-BC, Upper right side: movement of the proposed hybrid algorithm, Lower left hand side: zoomed movement of the original BB-BC, Lower right hand side: zoomed movement of the proposed hybrid algorithm.

The results logged in this chapter are obtained from 10000 random run for each test.

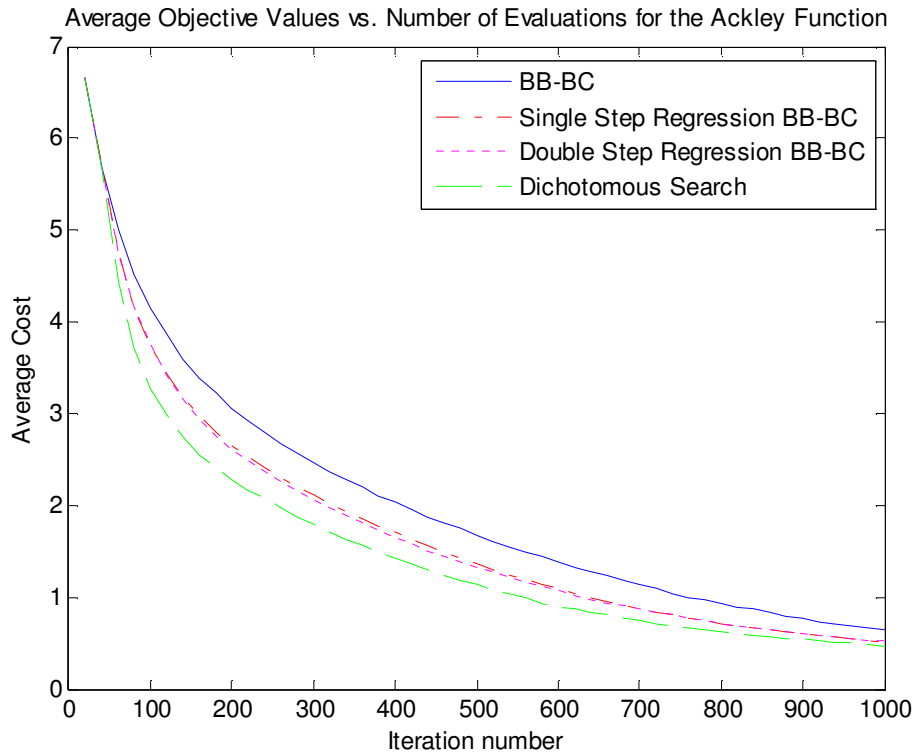
The following tables arranged with respect to the objective functions. For ellipsoid, step and sphere functions, the space topology is easier in comparison and the number of function evaluations (stopping criteria) is chosen to be half of the Ackley, Rastrigin or Rosenbrock counterpart.

**Table 3.5-Table 3.10** reports the cases for small search space and 20 individuals per population. **Figure 3.12-Figure 3.17** shows the average best fitness of whole runs with respect to the iteration number, respectively. On the other hand, **Table 3.11** and

**Figure 3.18** illustrate a condensed view for large search space and 30 individuals per population.

**Table 3.5** : Ackley test function, n = 20, 500 evaluations, search space: [-10, 10].

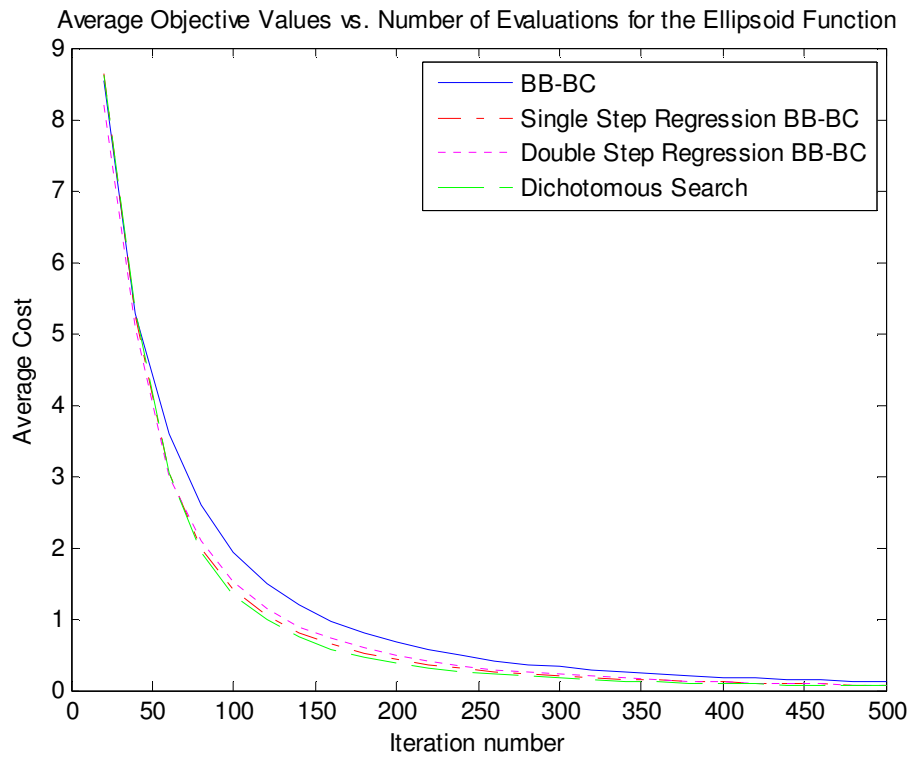
| ACKLEY                      | BB-BC | Single Step Regression | Double Step Regression | Dichotomous Search (OneStep) |
|-----------------------------|-------|------------------------|------------------------|------------------------------|
| Average Cost [improvement%] | 0.65  | 0.49[25%]              | 0.51[22%]              | 0.47[28%]                    |
| Median Cost [improvement%]  | 0.53  | 0.40[25%]              | 0.40[25%]              | 0.39[27%]                    |
| Std. Dev. Of Cost           | 0.43  | 0.36                   | 0.37                   | 0.33                         |



**Figure 3.12** : Improvement of the Ackley cost value with respect to increasing evaluation number.

**Table 3.6 :** Ellipsoid test function,  $n = 20$ , 500 evaluations, search space:  $[-10, 10]$ .

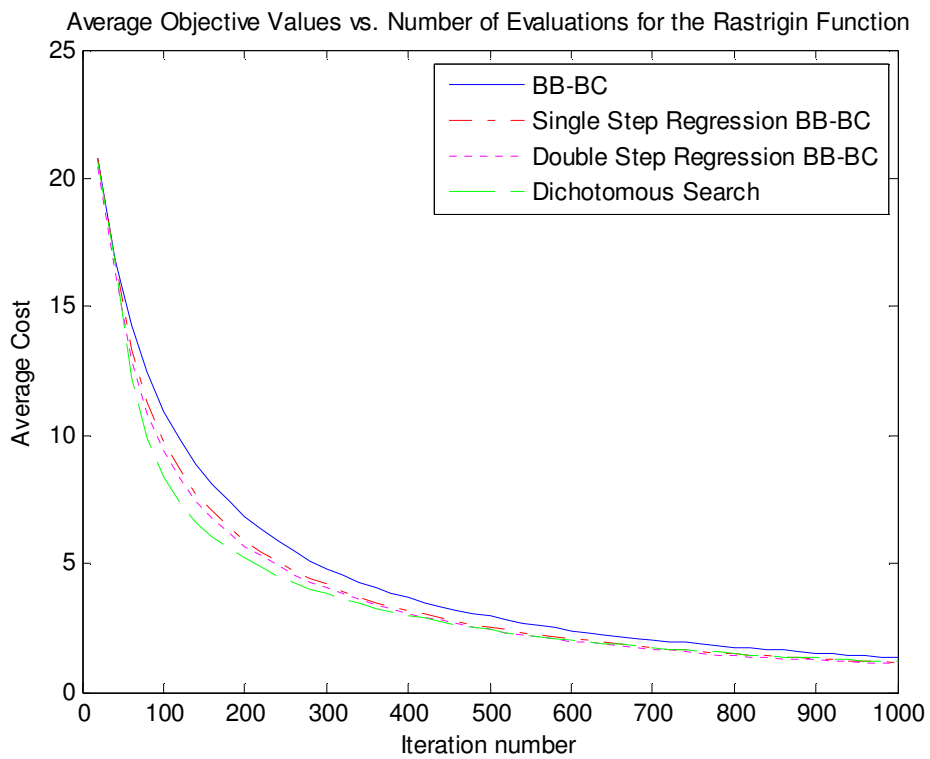
| ELLIPSOID                   | BB-BC | Single Step Regression | Double Step Regression | Dichotomous Search (OneStep) |
|-----------------------------|-------|------------------------|------------------------|------------------------------|
| Average Cost [improvement%] | 0.11  | 0.07[36%]              | 0.07[36%]              | 0.06[45%]                    |
| Median Cost [improvement%]  | 0.07  | 0.04[43%]              | 0.03[57%]              | 0.02[71%]                    |
| Std. Dev. Of Cost           | 0.10  | 0.08                   | 0.09                   | 0.07                         |



**Figure 3.13 :** Improvement of the Ellipsoid cost value with respect to increasing evaluation number.

**Table 3.7 :** Rastrigin test function,  $n = 20$ , 500 evaluations, search space:  $[-10, 10]$ .

| RASTRIGIN                   | BB-BC | Single Step Regression | Double Step Regression | Dichotomous Search (OneStep) |
|-----------------------------|-------|------------------------|------------------------|------------------------------|
| Average Cost [improvement%] | 1.35  | 1.15[15%]              | 1.11[18%]              | 1.18[13%]                    |
| Median Cost [improvement%]  | 1.26  | 1.11[12%]              | 1.09[13%]              | 1.15[9%]                     |
| Std. Dev. Of Cost           | 0.71  | 0.69                   | 0.70                   | 0.69                         |

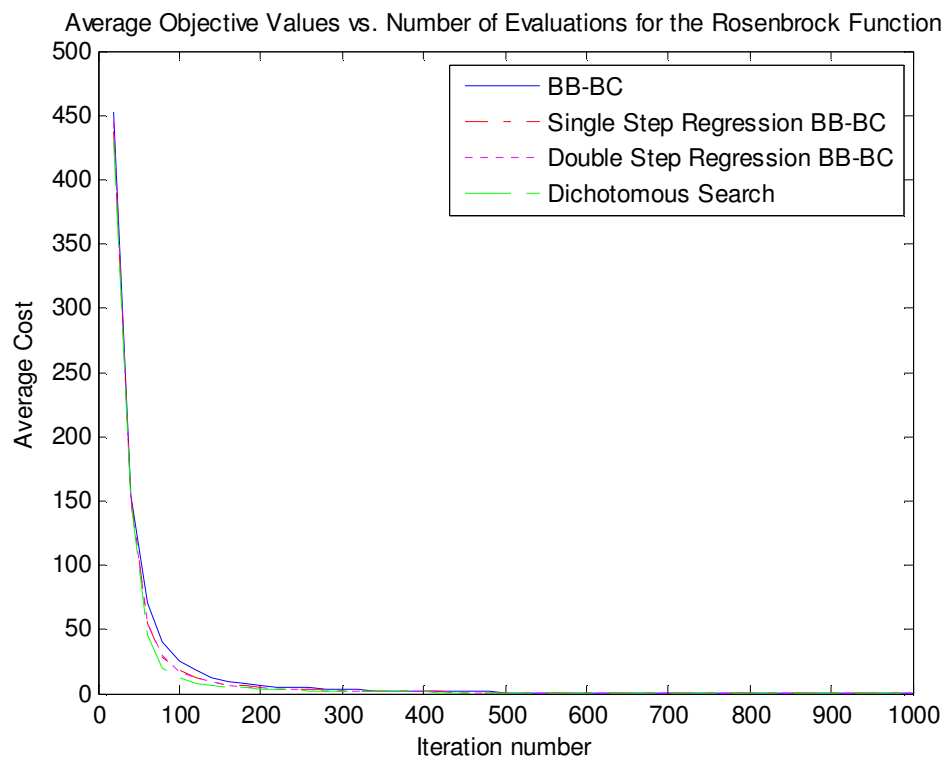


**Figure 3.14 :** Improvement of the Rastrigin cost value with respect to increasing evaluation number.



**Table 3.8 :** Rosenbrock test function,  $n = 20$ , 500 evaluations, search space:[-10, 10].

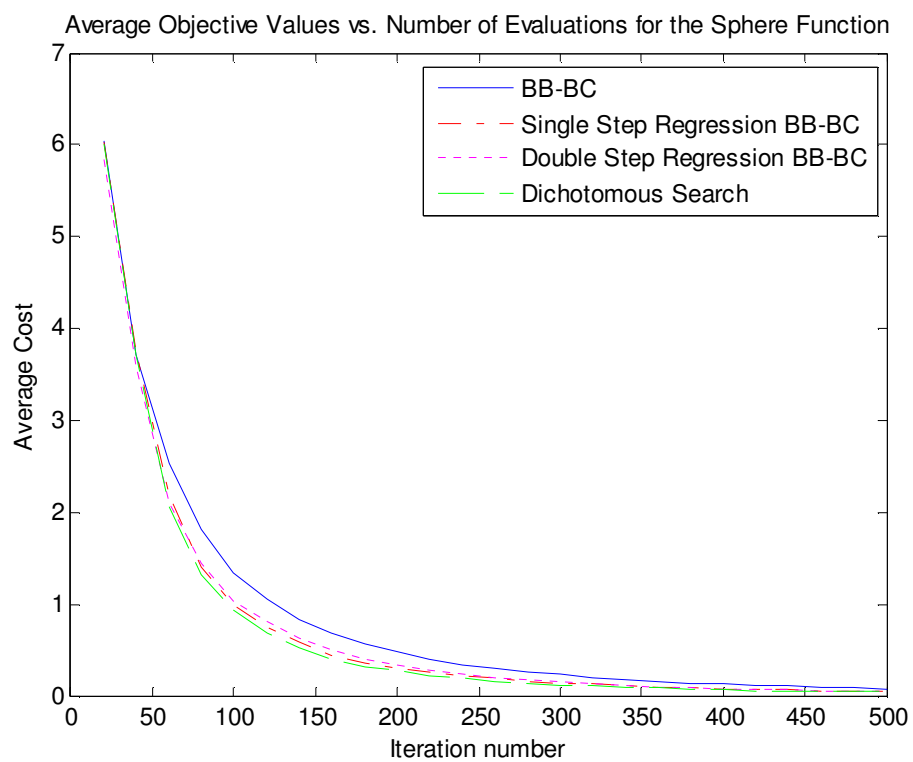
| ROSENBROCK                  | BB-BC | Single Step Regression | Double Step Regression | Dichotomous Search (OneStep) |
|-----------------------------|-------|------------------------|------------------------|------------------------------|
| Average Cost [improvement%] | 0.40  | 0.31[23%]              | 0.30[25%]              | 0.29[28%]                    |
| Median Cost [improvement%]  | 0.13  | 0.08[%38]              | 0.08[38%]              | 0.08[38%]                    |
| Std. Dev. Of Cost           | 0.95  | 0.78                   | 0.81                   | 0.79                         |



**Figure 3.15:** Improvement of the Rosenbrock cost value with respect to increasing evaluation number.

**Table 3.9 :** Sphere test function,  $n = 20$ , 500 evaluations, search space:  $[-10, 10]$ .

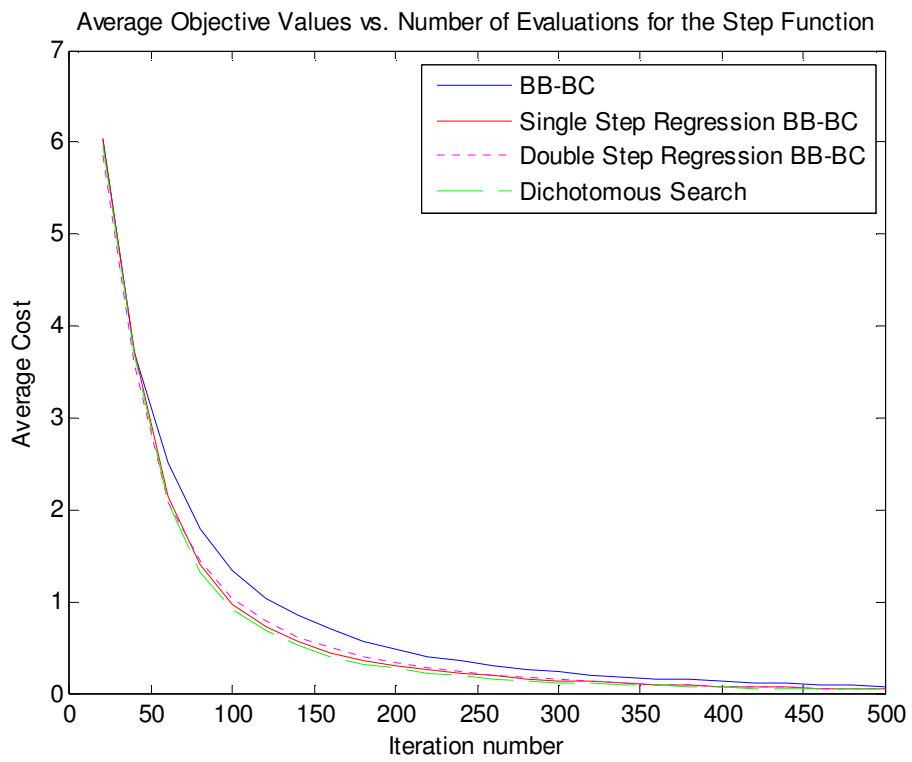
| SPHERE                      | BB-BC | Single Step Regression | Double Step Regression | Dichotomous Search (OneStep) |
|-----------------------------|-------|------------------------|------------------------|------------------------------|
| Average Cost [improvement%] | 0.08  | 0.05[38%]              | 0.05[38%]              | 0.04[50%]                    |
| Median Cost [improvement%]  | 0.05  | 0.03[40%]              | 0.03[50%]              | 0.02[60%]                    |
| Std. Dev. Of Cost           | 0.07  | 0.05                   | 0.06                   | 0.05                         |



**Figure 3.16 :** Improvement of the Sphere cost value with respect to increasing evaluation number.

**Table 3.10 :** Step test function,  $n = 20$ , 500 evaluations, search space:  $[-10, 10]$ .

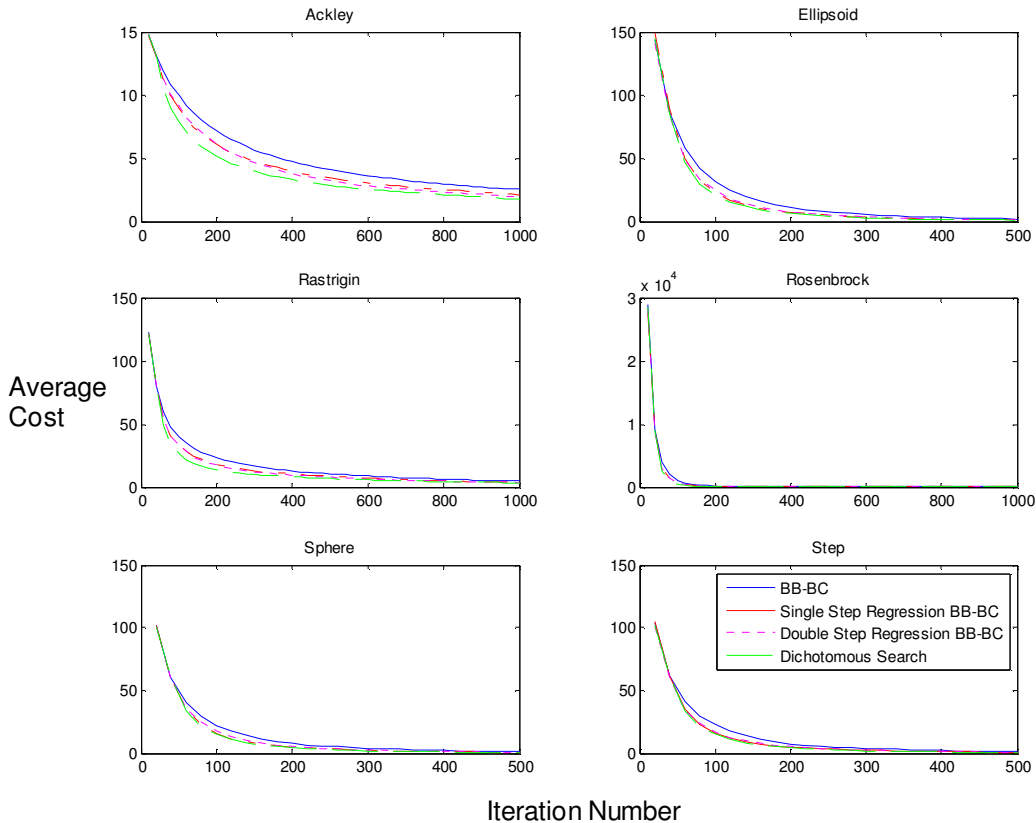
| STEP                        | BB-BC | Single Step Regression | Double Step Regression | Dichotomous Search (OneStep) |
|-----------------------------|-------|------------------------|------------------------|------------------------------|
| Average Cost [improvement%] | 0.08  | 0.05[38%]              | 0.05[38%]              | 0.04[50%]                    |
| Median Cost [improvement%]  | 0.05  | 0.03[40%]              | 0.03[40%]              | 0.02[60%]                    |
| Std. Dev. Of Cost           | 0.07  | 0.05                   | 0.06                   | 0.05                         |



**Figure 3.17 :** Improvement of the Rastrigin cost value with respect to increasing evaluation number.

**Table 3.11** : Average costs for all functions,  $n = 30$ , 3000 / 1500 evaluations, search space:  $[-50, 50]$ .

| Function   | BB-BC<br>Average<br>Cost | Single Step<br>Regression<br>Average Cost<br>[improvement%] | Double Step<br>Regression<br>Average Cost<br>[improvement%] | Dichotomous<br>Search Average<br>Cost<br>[improvement%] |
|------------|--------------------------|---|---|---|
| Ackley     | 1.14                     | 0.87[24%]   | 0.81[29%]   | 0.75[34%]   |
| Ellipsoid  | 0.33                     | 0.19[42%]   | 0.18[45%]   | 0.16[52%]   |
| Rastrigin  | 1.96                     | 1.63[17%]   | 1.53[22%]   | 1.65[16%]   |
| Rosenbrock | 3.59                     | 3.04[15%]   | 3.20[11%]   | 2.96[18%]   |
| Sphere     | 0.23                     | 0.13[43%]   | 0.12[48%]   | 0.11[52%]   |
| Step       | 0.24                     | 0.13[46%]   | 0.12[50%]   | 0.11[54%]   |



**Figure 3.18** : The improvements of the cost values for all functions for large search space through evaluations.

One can easily conclude that a slight improvement in the local search step can cause further improvement on total search performance by checking the second and last

columns of the tables. Main purpose is not to intensify on the local search step but the idea of amalgamating local and global search procedures. Better results could have been obtained by fine-tuning the parameters of the local search algorithm but generality of the hybrid algorithm would have been sacrificed.

The simulation results clearly illustrate the improvement on the algorithm performance. Though the new hybrid method is not faster or slower than the original BB-BC in time-wise, the accuracy achieved within the same number of fitness function evaluations is quite considerable and makes the routine worthy.

### **3.4 BB – BC with Local Directional Moves (BBBC – LS)**

#### **3.4.1 Algorithm formulation**

BBBC-LS has the same general algorithm run as in **Figure 3.6**. However, the algorithm defines new parameters to control crunching function and allowed number of function evaluations at the crunching phase.

The steps of the algorithm can be summarized as follows:

STEP 1: Form an initial generation of  $N$  individuals in a random manner.

STEP 2: Perform the crunching phase of the BB – BC algorithm. The centre of mass is selected as the fittest individual. This point becomes the first best point found in the iteration. Store this point.

STEP 3: Perform once more the consecutive banging and then crunching phases of the BB – BC algorithm. Crunching phase is switched to NM crunching after  $T_{fe}$  portion of function evaluations completed.

STEP 4: If the *best* point obtained in step 3 is better than the last stored point then this means an improvement then store that point. Next, generate a direction vector using one or two previous best candidate solution points so far attained and make  $n_h$  exploratory moves in that direction and assign a new virtual centre of mass on that direction if a better point has been obtained than the previous

fittest points. If the best (fittest) point remains the same after the local search phase then go straight to step 5.

STEP 5: Check the stopping criteria. If it is met stop; else go back to step 3.

Definition, abbreviation and value intervals for the algorithm specific parameters are listed in **Table 3.12**.

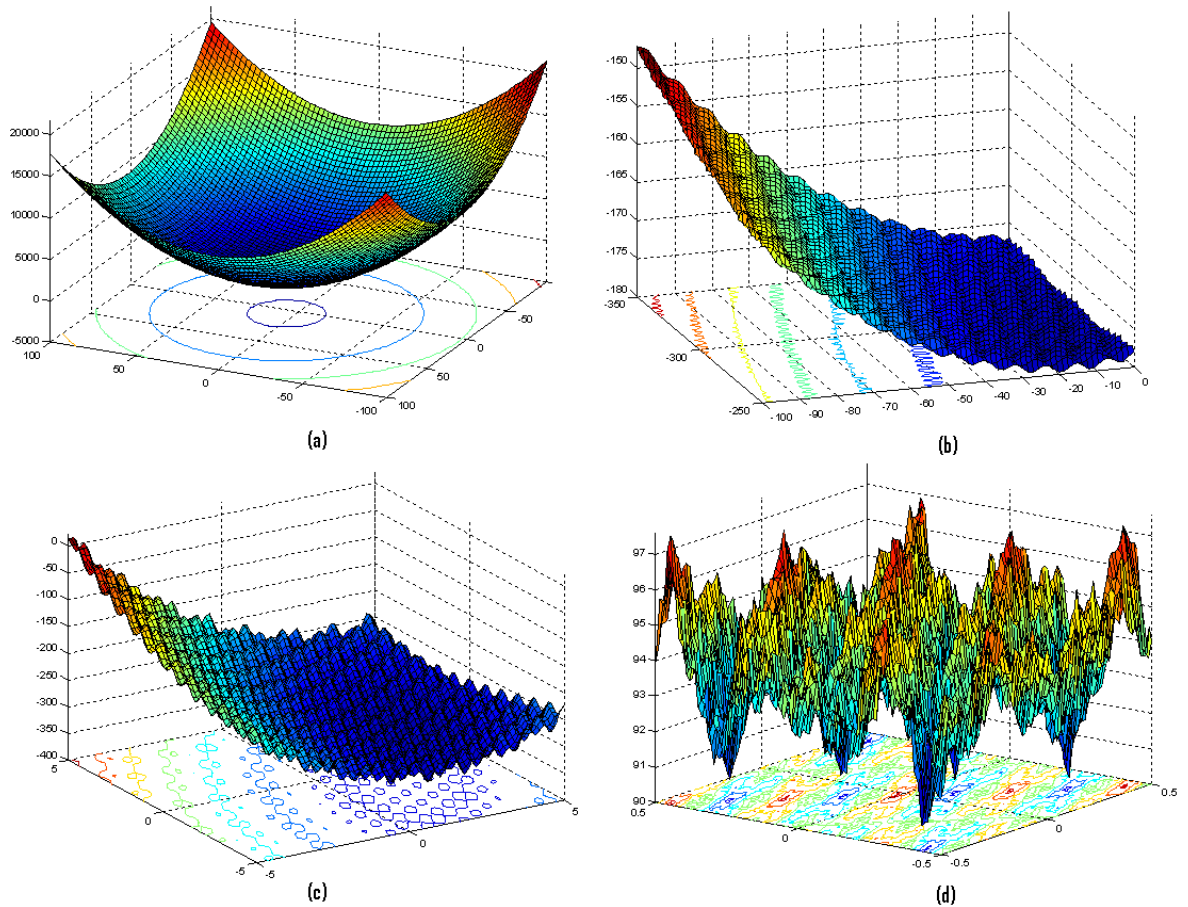
The proposed idea on this study is to speed up the search by checking some local points after the crunching phase of the main global search algorithm so as to maximize improvement probability. The neighboring points check procedure should be carried in the guided and limited direction(s). Otherwise, checking random neighbors or complete set of neighbors can cause unacceptable processing time or even search stagnation. In this study, the proposed local search moves of the hybridization procedure are based on defining a possible improving direction to check neighboring points. Between iterations, the movement of the best point forms a basis for linear search direction definition. Search directions are generated by utilizing auto regression on the locations of the representative points of consecutive crunch phases. The local search operation can be performed for a few predetermined numbers of steps on these directions so abstaining from sticking into a local optimum point. Any local search method can be utilized in these generated directions; here, expansion and contraction moves of basic simplex search method and dichotomous search algorithms are exploited for the local search phase of hybridized optimization method.

**Table 3.12** : Definition of algorithm parameters.

| Abb.     | Definition   | Value Interval   | Data Type |
|----------|--|--|-----------|
| $N$      | population size  | $N \geq D + 1$ , if NM crunching is used                 | integer   |
| $nm_b$   | allowed function evaluation budget for the NM crunching phase (if used)  | $nm_b < \text{total function evaluation budget (FE)}$    | integer   |
| $nm_t$   | NM crunching tolerance error (if used), (algorithm ends either $nm_b$ or $nm_t$ fulfilled)                           | $0 < nm_t < \infty$                                      | double    |
| $n_h$    | number of expansion/contraction steps performed between each iteration   | $0 < n_h < \text{total function evaluation budget (FE)}$ | integer   |
| $T_{fe}$ | Normalized crunching phase switching parameter: After $T_{fe}$ proportion of total FE carried switch to NM crunching | $0 \leq T_{fe} \leq 1$                                   | double    |
| $sm$     | explosion strength adjusting parameter, determines mean step size of banging phase                                   | $1 \leq sm \leq \infty$                                  | double    |

### 3.4.2 Simulation results for BBBC-LS

To evaluate the performance of the newly proposed hybrid method (BBBC-LS), the algorithm is applied to four test functions with distinct characteristics, selected from the benchmark test bed proposed for the CEC'05 Special Session on Real-Parameter Optimization (Suganthan et al, 2005). Three-dimensional mappings for two dimensional search spaces of the selected benchmark functions are given in **Figure 3.18**. In the simulations, 10 dimensional versions of these functions are used. Mathematical expressions, search range, global minimum function values and basic properties for the benchmark functions are given in **Table 3.13**.



**Figure 3.19** : Benchmark test functions from CEC'05 competition. (a) Shifted Sphere, (b) Shifted Rotated Griewank, (c) Shifted Rotated Rastrigin, (d) Shifted Rotated Weierstrass.

To verify the effectiveness of the proposed approach, three well known optimization routines are utilized on the same test functions: Genetic Algorithm (GA) is probably the most commonly accepted umbrella term covering many variants. Covariance Matrix Adaptation Evolutionary Strategies (CMA-ES) and Particle Swarm Optimization (PSO) have also been successfully applied in many research and application areas over the past few decades. For comparison purposes, GA is used as implemented in the Global Optimization Toolbox of Matlab R2010a; CMA-ES is used as detailed in (Hansen et al, 2003; Hansen, 2006; Hansen et al, 2009) and code is used as the January, 2011 version in Hansen's web page (Url-1); and PSO Toolbox (Url-2) is used for particle swarm evaluations.



**Table 3.13** : Summary of the benchmark functions (D = 10).

| Function   | Minimum         | Search Range          | Properties  |
|--|-----------------|-----------------------|---|
| <p>Shifted Sphere:</p> $F_{sphere} = \sum_{i=1}^D z_i^2 + bias_1$ <p><math>z=x-o</math>, <math>x=[x_1, x_2, \dots, x_D]</math></p>   | $bias_1 = -450$ | $x \in [-100, 100]^D$ | <ul style="list-style-type: none"> <li>-Unimodal</li> <li>-Shifted</li> <li>-Separable</li> <li>-Scalable</li> </ul>  |
| <p>Shifted Rotated Griewank:</p> $F_{griewank} = \sum_{i=1}^D \frac{z_i^2}{4000} - \prod_{i=1}^D \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + bias_2$ <p><math>z=(x-o)*M</math>, <math>x=[x_1, x_2, \dots, x_D]</math>,<br/> <math>M = M'(1+0.3 N(0, 1) )</math><br/> <math>M'</math>: Linear Transformation Matrix with condition number 3</p>   | $bias_2 = -180$ | $x \in [-500, 0]^D$   | <ul style="list-style-type: none"> <li>-Multi-modal</li> <li>-Rotated</li> <li>-Shifted</li> <li>-Non-Separable</li> <li>-Scalable</li> </ul>   |
| <p>Shifted Rotated Rastrigin:</p> $F_{rastrigin} = \sum_{i=1}^D (z_i^2 - 10 \cos(2\pi z_i) + 10) + bias_3$ <p><math>z=(x-o)*M</math>, <math>x=[x_1, x_2, \dots, x_D]</math>,<br/> <math>M</math>: Linear Transformation Matrix with condition number 2</p>   | $bias_3 = -330$ | $x \in [-5, 5]^D$     | <ul style="list-style-type: none"> <li>-Multi-modal</li> <li>-Rotated</li> <li>-Shifted</li> <li>-Non-Separable</li> <li>-Scalable</li> <li>-Huge Number of Local Optima</li> </ul>                                 |
| <p>Shifted Rotated Weierstrass:</p> $F_{weierstrass} = \sum_{i=1}^D \left( \sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (z_i + 0.5))] \right) - D \sum_{k=0}^{k_{max}} [a^k \cos(2\pi b^k 0.5)] + bias_4$ <p><math>a=0.5</math>, <math>b=3</math>, <math>k_{max}=20</math>,<br/> <math>z=(x-o)*M</math>, <math>x=[x_1, x_2, \dots, x_D]</math>,<br/> <math>M</math>: Linear Transformation Matrix with condition number 5</p> | $bias_4 = 90$   | $x \in [-0.5, 0.5]^D$ | <ul style="list-style-type: none"> <li>-Multi-modal</li> <li>-Rotated</li> <li>-Shifted</li> <li>-Non-Separable</li> <li>-Scalable</li> <li>-Continuous</li> <li>-Differentiable only on a set of points</li> </ul> |

**Table 3.14** reports the benchmark function scores for all algorithms at the end of 500 / 1000 / 2000 function evaluations (FE) for 1000 independent runs. In these simulations, BBBC-LS algorithm uses the following parameters:  $N = 15$ ,  $n_h = 2$ ,  $T_{fe} = 0.6$ ,  $sm = 10$ ,  $nm_t = 0.1 / (1 + k / sm)$  where  $k$  is the iteration number. As local

directional move, dichotomous search is used. Stopping criterion for the NM crunching phase is chosen to be termination tolerance, therefore NM crunching budget ( $nm_b$ ) parameter set to infinity. The other algorithms are optimized only for population size; all the remaining algorithm specific parameters are either left as default / suggested parameters or self tuned by the algorithm itself.

**Table 3.14** : Average performance scores.

|                |                 |                 |                 |
|----------------|-----------------|-----------------|-----------------|
| Sphere         | FE: 500         | FE: 1000        | FE: 2000        |
| GA             | -433,502        | -445,616        | -448,277        |
| CMA-ES         | -449,688        | -449,997        | -450,000        |
| PSO            | -422,404        | -441,127        | -449,009        |
| <b>BBBC-LS</b> | <b>-448,667</b> | <b>-449,838</b> | <b>-449,979</b> |
| Rastrigin      | FE: 500         | FE: 1000        | FE: 2000        |
| GA             | -231,448        | -273,896        | -276,315        |
| CMA-ES         | -246,733        | -274,883        | -309,702        |
| PSO            | -230,444        | -262,460        | -287,820        |
| <b>BBBC-LS</b> | <b>-299,244</b> | <b>-309,962</b> | <b>-310,070</b> |
| Griewank       | FE: 500         | FE: 1000        | FE: 2000        |
| GA             | -53,481         | -99,3917        | -116,699        |
| CMA-ES         | -169,161        | -169,982        | -170,983        |
| PSO            | -167,573        | -173,789        | -179,555        |
| <b>BBBC-LS</b> | <b>-174,719</b> | <b>-177,160</b> | <b>-178,314</b> |
| Weierstrass    | FE: 500         | FE: 1000        | FE: 2000        |
| GA             | 101,271         | 99,491          | 99,510          |
| CMA-ES         | 101,640         | 97,047          | 94,414          |
| PSO            | 100,386         | 98,850          | 98,866          |
| <b>BBBC-LS</b> | <b>98,316</b>   | <b>97,020</b>   | <b>95,223</b>   |

**Table 3.15** and **Table 3.16** serve for summarizing the performances of the algorithms: In **Table 3.15**, every entry gives the order for the corresponding algorithm at the end of corresponding FE budget. **Table 3.16** reports the number of being the best method on 4 test functions and 3 different FE levels (summing up 12 cases) and assigns an overall rating considering the mean place.

The power of BBBC-LS lies not only in its capability for quick convergence but also in its low level of complexity. There are a few number of parameters to be tuned: the user should select the population size ( $N$ ), number of expansion/contraction steps ( $n_h$ ), NM crunching budget ( $nm_b$ ), NM crunching tolerance ( $nm_t$ ), crunching phase switching parameter ( $T_{fe}$ ) and the explosion strength adjusting parameter ( $sm$ ). However, GA, CMA-ES and PSO have many parameters to be selected by the

designer, though many variants of these methods generally offer self selection / adaptation of these parameters settings.

**Table 3.15 :** Order of algorithms (1: Best, 2: Second, 3: Third, 4: Worst).

| Sphere         | FE: 500  | FE: 1000 | FE: 2000 | Griewank       | FE: 500  | FE: 1000 | FE: 2000 |
|----------------|----------|----------|----------|----------------|----------|----------|----------|
| GA             | 3        | 3        | 4        | GA             | 4        | 4        | 4        |
| CMA-ES         | 1        | 1        | 1        | CMA-ES         | 2        | 3        | 3        |
| PSO            | 4        | 4        | 3        | PSO            | 3        | 2        | 1        |
| <b>BBBC-LS</b> | <b>2</b> | <b>2</b> | <b>2</b> | <b>BBBC-LS</b> | <b>1</b> | <b>1</b> | <b>2</b> |
| Rastrigin      | FE: 500  | FE: 1000 | FE: 2000 | Weierstrass    | FE: 500  | FE: 1000 | FE: 2000 |
| GA             | 3        | 3        | 4        | GA             | 3        | 4        | 4        |
| CMA-ES         | 2        | 2        | 2        | CMA-ES         | 4        | 2        | 1        |
| PSO            | 4        | 4        | 3        | PSO            | 2        | 3        | 3        |
| <b>BBBC-LS</b> | <b>1</b> | <b>1</b> | <b>1</b> | <b>BBBC-LS</b> | <b>1</b> | <b>1</b> | <b>2</b> |

**Table 3.16 :** Summary of algorithm comparison.

| Algorithm      | # of first rankings | Average ranking | Overall rank |
|----------------|---------------------|-----------------|--------------|
| GA             | 0                   | 3.5833          | 4            |
| CMA-ES         | 4                   | 2               | 2            |
| PSO            | 1                   | 3               | 3            |
| <b>BBBC-LS</b> | <b>7</b>            | <b>1.4167</b>   | <b>1</b>     |

There are many metrics on algorithm complexity but neither of them is universally accepted. In CEC'05, running time difference between 200000 function evaluations ( $T1$ ) and the complete computing time for the algorithm with 200000 function evaluations ( $T2$ ) have been normalized with a run time of reference mathematical function ( $T0$ ) on a dedicated computer. This can be formulated as in (3.8),

$$Complexity = \frac{\langle T2 \rangle - T1}{T0} \quad (3.8)$$

where  $\langle . \rangle$  symbol stands for averaging function over multiple runs. The details for the complexity analysis can be further investigated on Suganthan et al. (2005). The results for complexity analysis can be found in **Table 3.17**. The test function used for complexity analysis is randomly chosen to be as the shifted rotated Weierstrass function.

**Table 3.17 : Complexity Analysis.**

| <b>Algorithm</b> | <b>CEC–2005 Complexity</b> | <b>Run Time (s)</b> | <b>Number of Parameters to Adjust</b> |
|------------------|----------------------------|---------------------|---------------------------------------|
| GA               | 27.9551                    | 189.6874            | 13                                    |
| CMA–ES           | 44.9625                    | 294.4760            | 24                                    |
| PSO              | 25.1343                    | 172.3075            | 14                                    |
| <b>BBBC–LS</b>   | <b>12.8308</b>             | <b>96.5014</b>      | <b>6</b>                              |

### 3.5 Conclusion

A simple but effective hybridization procedure for the Big Bang–Big Crunch optimization algorithm is presented in this chapter. The method generates a direction vector from the past positions of the best individuals found so far and investigates on this line with extraction or contraction moves. This local search phase is modular and works without interception to the original BB – BC algorithm. Moreover, the crunching phase of the algorithm is expanded to include a simplex based approach; namely, the Nelder– Mead optimization method.

The crunching phase using the Nelder – Mead optimization method improves the exploitation capability of the BB–BC algorithm so, it is more appropriate to use it towards the final steps of the search. Therefore, the proposed method introduces a switching parameter ( $T_{fe}$ ) for crunching phase selection. Then, at the early iterations, weighted mean of the candidate member solutions or the best solution member is selected as the centre of mass; whereas, after the switching condition is fulfilled, NM crunching is used for more exploitive search. The switching threshold parameter is assigned at the beginning and kept constant throughout the search, but it is a promising idea to adapt this parameter in a dynamical manner. This adaptation could be performed based on a feedback controller observing the population diversity and history of the population diversity.

The simulation results on various test functions clearly illustrate the superiority of adding local directional moves over the original BB – BC algorithm. The accuracy achieved by the newly proposed method within the same number of fitness function evaluations is quite considerable and makes this routine worthy. Moreover, as a

compact new algorithm, BBBC-LS turns out to be a good alternative to the widely accepted state-of-the-art evolutionary optimization algorithms. Its accuracy is better or at least comparable for the tested benchmark functions and the complexity and running time are far better than GA, CMA-ES and PSO.



## **4. SINGLE LEAP-BIG BANG BIG CRUNCH OPTIMIZATION APPROACH TO SINGLE OBJECTIVE AIRPORT GATE ASSIGNMENT PROBLEM**

### **4.1 Introduction**

The air transportation becomes more and more widespread during the past fifteen years. As well as the opportunity of travelling long distances in reasonable short time duration, the moderate prices due to competition of the companies made several travelers to choose airline industry. These facts tremendously increased the traffic in the airports compared to mid-1990s. In addition, the hub-and-spoke system has resulted in a large volume of baggage and passengers transferring between flights (Bazargan, 2004). Assigning arriving flights to airport gates is therefore an important issue in daily operations of an airline. It has a major impact on maintaining the efficiency of flight schedules, passenger satisfaction and the revenue obtained.

The problem of finding a suitable gate assignment is generally handled in three levels. In the first level, the ground controllers use the flight schedule to examine the capacity of the gates to accommodate these flights. The second level involves the development of daily plans before the actual day of operation. In the third level, because of the unexpected situations such as delays, bad weather, mechanical failure and maintenance requirements, these daily plans are updated and revised on the same hour/day of the operation (Bolat, 2000). In this chapter, the second and the third levels of operation are considered.

Possible objective functions can be defined in terms of the staying time of the planes in the gates, number of passengers in aircrafts, the total walking distances belonging to the passengers of all scheduled flights within a specified and closed time interval. Therefore, the problem formulation can vary quite a lot due to this large span of objectives. Moreover, basic gate assignment problem is NP-hard (non-deterministic polynomial-time hard) (Obata, 1979) quadratic assignment problem. Because of these, there are various approaches to this problem in the literature with respect to requirements imposed. The solution approaches have two heavily interacting main branches: rule based expert systems and mathematical models. In the implementation

given in this chapter, the GAP objective is to maximize total gate time as an integer programming mathematical formulation that uses multiple time slots and the basic constraint that allows one flight at one gate at one time. No rule-based expert system is utilized algorithm; but in the system developed for Atatürk Airport, the constraints are processed by user-defined rules.

Teodorovic & Guberinic (1984) and Teodorovic & Stojkovic (1990) focus on total passenger delay and the number of flights cancellations in the case of irregularity of flights. Among other possible criteria, passenger walking distances (Hu & Paulo, 2007; Ding et al, 2004; Ding et al, 2004; Ding et al, 2005; Haghani & Chen., 1998; Babic et al, 1984; Wirasinghe & Bandara, 1990; Bandara & Wirasinghe, 1992); baggage transfer distances (Hu & Paulo, 2007; Haghani & Chen., 1998) are also considered. Chang (1994) considers the distance covered by passengers in carrying their baggage as an objective in addition to passenger walking distance. Even any objective criterion has factions in implementation: for example, passenger walking distance can be handled as,

- I. minimize the sum of total distance that all passengers walk,
- II. minimize the distance after baggage claim area,
- III. minimize connection flight travelling distance,
- IV. minimize the maximum distance that a passenger need to walk
- V. minimize the number of passengers that need to walk more than x units.

The list can be further extended. Unfortunately, assignment objectives depending on passenger walking distance are quite fragile (Dorndorf et al, 2007).

Genetic Algorithms (GAs) are the most known and widespread used global optimization methods. Since GAs use random number generators and they exhibit an ability to avoidance to get trapped to local optima they are considered to be successful search procedures when the objective function is nonlinear, non-derivative and discontinuous. Some researchers proposed GA based methods for the gate assignment problem (Gu & Chung, 1999; Hu & Paulo, 2007; Bolat, 2001). All the approaches utilizing population based routines, including GA based approaches, use global optimization methods to top down solve the problem or to improve the result of some heuristics. However, forming a complete solution candidate or altering the list once all the flights are assigned can be quite tardy for GA or similar GA like



stochastic methods since they oblige to check all constraints to build up a valid solution. Therefore, using stochastic methods to ameliorate assignment after all the list has been built up is not a good solution alternative for GAP in practical applications.

For the İstanbul Atatürk Airport's operator, criterion of highest priority is to increase the revenue obtained from the gate allocation operation. The most important parameter in the revenues is therefore the allocated gates, which are available in a limited number. The more efficiently the gates are assigned to the aircrafts, the lesser idle time is left between two successive flights and this means that more passengers use the gates. Hence, the revenue and passenger satisfaction are both increased. Flight gates are the major items addressed in the GAP. At İstanbul Atatürk Airport, as well as the most of the airports throughout the world, the revenues are majorly dependent upon assignment of an airplane to a gate or not. This leads to a cost function which changes greatly if an airplane is assigned to a gate or not. This gives rise to a discontinuous objective function or more generally, a cost or fitness function where inter-gate aircraft switches do not have a great influence on it.

Next section gives the mathematical description of the problem. The details of the proposed method are presented in section 3, the simulation results are given in section 4. The developed system for the İstanbul Atatürk Airport this airport is given in section 5. The concluding remarks of the chapter are finally given in section 6.

## **4.2 Problem Formulation**

The objective is to maximize gate duration, which is total time of the gates allocated for all flights of a day. The basic constraint of the GAP imposed in the formulation can be stated as follows: one gate can only accommodate a single aircraft at a time and that therefore two flights must not be assigned to the same gate if their staying times overlap in time (Dorndorf et al, 2007). To measure density of the gates, the whole day is sampled for  $n$  minutes, where  $n$  can be chosen as 5 or 10 in a practical application. Note that selection of the length of a time slot directly effects the algorithm run time. In literature, selected time slots are in between five minutes and one hour duration (Bolot, 1999; Bolot 2001; Haghani & Chen., 1998). This time interval corresponding to  $n$  minutes is called a time slot and the density is measured by counting allocated timeslots.

The parameters related to gate assignment problem are defined as follows:

$N$ : number of aircrafts,

$N_g$ : number of gates,

$N_{oa}$ : number of open air parking places

$N_s$ : number of stands where  $N_s = N_g + N_{oa}$

$N_t$ : number of time slots in a day (depends on time slot length  $n$ ,  $N_t = 24*60/n$ )

$T_{A(i)}$ : arrival time of flight  $i$ ,

$T_{D(i)}$ : departure time of flight  $i$ ,

$M_u$ : ( $N \times N_t$ ) matrix of aircrafts (scheduling) where,

$M_u(i,j) = 1$ , if the aircraft  $i$  is at the airport in time slot  $j$  according to  $T_{A(i)}$  and

$T_{D(i)}$ ,

$M_u(i,j) = 0$ , if otherwise.

$M_c$ : ( $N_s \times N_t$ ) matrix of assignments (gate assignments) where,

$M_c(i,j) = U$ , ( $U=1,\dots,N$ ), if the gate  $i$  is assigned at time slot  $j$  to the  $U^{th}$  flight,

$M_c(i,j) = 0$ , if otherwise.

The function to be maximized can be formulated as in **(4.1)**,

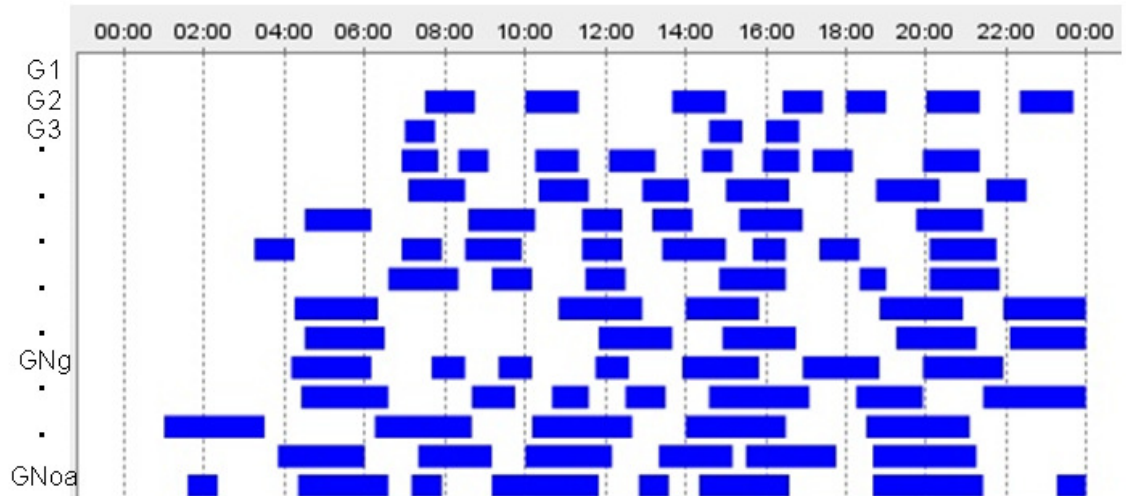
$$F_{fitness} = \sum_{k=1}^{N_g} \sum_{l=1}^{N_t} any(M_c(k,l)) \quad (4.1)$$

where,

$any(M_c(k,l)) = 1$ , if  $M_c(k,l) \neq 0$ ;

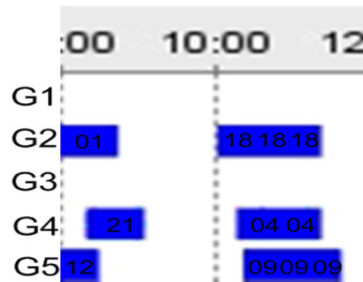
$any(M_c(k,l)) = 0$ , if otherwise.

**Figure 4.1** illustrates an assignment list for the planes. The vertical axis represents the gates available and the horizontal axis is the time. The list is given for a whole day. The planes, depicted as horizontal bars, are shown to occupy the corresponding gates for certain sojourn.



**Figure 4.1 :** A sample gate allocation.

**Figure 4.2** illustrates certain focused area of Figure xxx; that is, the assigned planes to the first 5 gates for the time interval of 8am to 10am. In this specific interval, first gate has no assigned planes; whereas, second gate resides the plane *01* for one time slot and the plane *18* for three time slots. Time axis is displayed in discrete version where each day has been divided into  $(24*60/n)$  timeslots.



**Figure 4.2 :** A sample gate allocation – condensed view.

### 4.3 Heuristic and Optimization Based Solution Approaches

In this section, firstly a greedy method from the literature will be introduced. Secondly, a new heuristic method that has been named as ground time duration maximization algorithm (GTMA) will be discussed. Finally, one of the main contributions of this thesis work, implementation of the Single Leap-Big BangBig Crunch (SL-BBBC) method will be given.

The design of an efficient heuristic becomes a paramount importance and constitutes the key focus of this important application (Xu & Bailey, 2001). Deterministic solutions provide a good initial starting point for the stochastic algorithm. Starting with the best heuristic solution, the stochastic approaches can improve the solution by modifying the assignment list that is given in **Figure 4.1**. On contrary to all the previous work so far done in this area, the newly proposed method does not work on the final assignment list, but on the plane ordering process. Plane ordering process can be defined as assigning priority for all the planes with respect to a chosen criterion. Once all the planes are ordered, they are tried to be allocated starting from the one having highest priority. By doing so, all the constraint satisfaction checks needed after a modification on the assignment list can be omitted. Besides this, new approach can be used with any heuristic that constitutes a basis for ordering – or priority assignment – for the flights and with the allocation module in any airport having different constraints.

### **4.3.1 Heuristic approaches**

#### **4.3.1.1 A previously reported heuristic: Greedy algorithm for minimizing the number of flights assigned to the apron**

In their previous works, Ding et al. (2004a, 2004b, 2005) proposed a greedy algorithm to minimize the number of the ungated flights. The flights are ordered with respect to departure times and assigned to the gates one by one respecting this order. If there are no gates available, then that flight is assigned to the apron. The algorithm steps are summarized for quick referencing as below:

STEP 1: Sort the flights according to the departure time  $T_{D(i)}$ .

STEP 2: Set  $g_k = -1$  for all gates where  $g_k$  ( $1 < k < Ng$ ) represents the earliest available time in

gate-k (that is the departure time of the last assigned plane to gate-k).

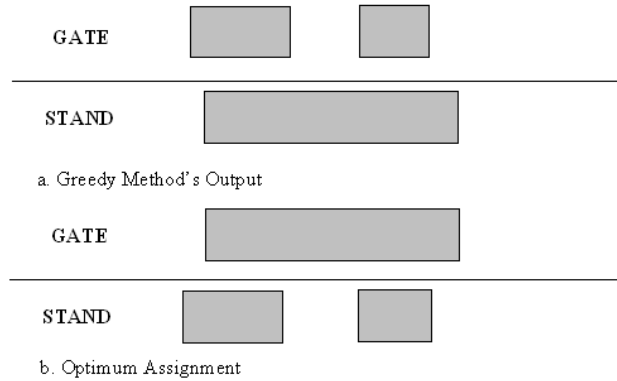
STEP 3: For each flight  $i$  find gate-k such that  $g_k < T_{A(i)}$  and  $g_k$  is maximized

(i) if such  $k$  exists, assign flight  $i$  to gate-k, update  $g_k = T_{D(i)}$ .

(ii) else assign flight  $i$  to the apron.

STEP 4: Output the result.

The algorithm is proven optimum for minimizing the number of flights assigned to the apron but it has a weakness in maximizing the total gate time if the early departing flight is a short staying one as illustrated in **Figure 4.3**.



**Figure 4.3** : Failure of greedy method.

#### 4.3.1.2 A new heuristic approach: Ground time duration maximization algorithm (GTMA)

GTMA is designed with the objective to maximize the total gate duration. The underlying idea is to sort planes with respect to their staying durations and then allocating them one by one. That is to say, the longest staying plane is assigned with the highest priority:

STEP 1: Pick the flight with longest time interval between its arrival and departure,





STEP 2: Start from gate #1,

STEP 3: Assign the flight to the gate if possible; else, select the next gate and repeat the procedure until finding a vacant gate.

STEP 4: Remove the flight from the list once it is assigned.

STEP 5: Go to step #1 until all the flights have been assigned.

This heuristic method generates an order for the allocation process as the greedy method. The long staying flights are assigned in the first place and the flight with smaller gate durations can be inserted in between these larger gate durations. However, this method may not be optimal for certain cases with respect to gate time maximization criterion as illustrated in **Figure 4.4**.

|                           |       |  |
|---------------------------|-------|--|
| <b>Optimum Assignment</b> | Gate  |  |
|                           | Apron |  |
| <b>GTMA Assignment</b>    | Gate  |  |
|                           | Apron |  |

**Figure 4.4 :** Illustration of failure of GTMA.

#### 4.3.2 Single Leap-Big Bang Big Crunch algorithm (SL- BBBC)

All the studies in GAP that are based on evolutionary algorithms focus on modifying the final assignment list given in **Figure 4.1** in different ways. This makes the running procedure highly nonlinear and that causes very long run time for the algorithm. This is unfavorable or unacceptable in most cases because frequently occurring delays in the flights pin down a quick reconfiguration of the gate assignment list.

Single Leap-Big Bang Big Crunch (SL-BBBC) algorithm makes its progress on an individual which is initially assigned by the deterministic solution developed by any heuristic plane ordering algorithm. SL-BBBC algorithm is used after the heuristic GTMA since it provides much better results compared to greedy heuristic method reported in Section 4.3.1.1. In Single Leap-Big Bang Big Crunch (SL-BBBC) algorithm, there is no population of solutions, so no information exchange between solution candidates will take place. For this reason, the BBBC algorithm has been renamed as —Single Leap-Big Bang Big Crunch. The unique solution at hand is modified at each iteration step and if a better solution is attained, then the next iteration works on newly generated solution. In summary, aforementioned GTMA algorithm is used to find deterministic solution and then the solutions are further improved by using SL-BBBC, that is to say, the deterministic algorithms serve initial point for the evolutionary algorithm.

Key point is that the SL-BBBC algorithm works on the assignment order of planes instead of the final assignment list itself. Once the initial assignment list and the



The new assignment methodology can be summarized as follows:


STEP 1: Apply GTMA and find an assignment list.

STEP 2: Log the order in which the planes are assigned.

STEP 3: Apply SL-BBBC to find better assignment list.

SL-BBBC algorithm can be implemented in three possible formulations.


- a. Interchanging the order of only two flights with random distances away from a random center in the ordering list: In **Figure 4.6**, the center is chosen to be the position of Flight #3 and the distance is chosen to be two units. Then Flight #1 and Flight #5 interchange the positions. Note that the distance here is related with the explosion strength and center is related with the center of mass in the original BB-BC algorithm (Erol & Eksin, 2006).

| Original Order |  | Result of one BBBC iteration |
|----------------|---|------------------------------|
| Flight #1      | <b>RE - ORDERING</b>  | Flight #5                    |
| Flight #2      |   | Flight #2                    |
| Flight #3      |   | Flight #3                    |
| Flight #4      |   | Flight #4                    |
| Flight #5      |   | Flight #1                    |
| Flight #6      |   | Flight #6                    |
| Flight #7      |   | Flight #7                    |

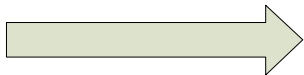
**Figure 4.6** : Reordering of the flights: Type-a reordering.

- b. Randomly permuting the flights in between randomly selected two points. In **Figure 4.**, all the flights between Flight #2 and Flight #6 are reordered. Number of flights to be rearranged is correlated with the explosion strength.
- c. Interchanging the order of N random flights pairs with random distances away from random centers in the list. This is a generalized version for case (a). Here, both the number of changes and the distances in between are related with the explosion strength that is getting smaller as the number of iterations increase (**Figure 4.**).



| Original Order |  | Result of one BBBC iteration |
|----------------|---|------------------------------|
| Flight #1      | <b>RE - ORDERING</b>  | Flight #1                    |
| Flight #2      |   | Flight #4                    |
| Flight #3      |   | Flight #3                    |
| Flight #4      |   | Flight #6                    |
| Flight #5      |   | Flight #2                    |
| Flight #6      |   | Flight #5                    |
| Flight #7      |   | Flight #7                    |

**Figure 4.7 :** Reordering of the flights: Type-b reordering.

| Original Order |  | Result of one BBBC iteration |
|----------------|---|------------------------------|
| Flight #1      | <b>RE - ORDERING</b>  | Flight #5                    |
| Flight #2      |   | Flight #2                    |
| Flight #3      |   | Flight #4                    |
| Flight #4      |   | Flight #3                    |
| Flight #5      |   | Flight #1                    |
| Flight #6      |   | Flight #6                    |
| Flight #7      |   | Flight #7                    |

**Figure 4.8 :** Reordering of the flights: Type-c reordering.

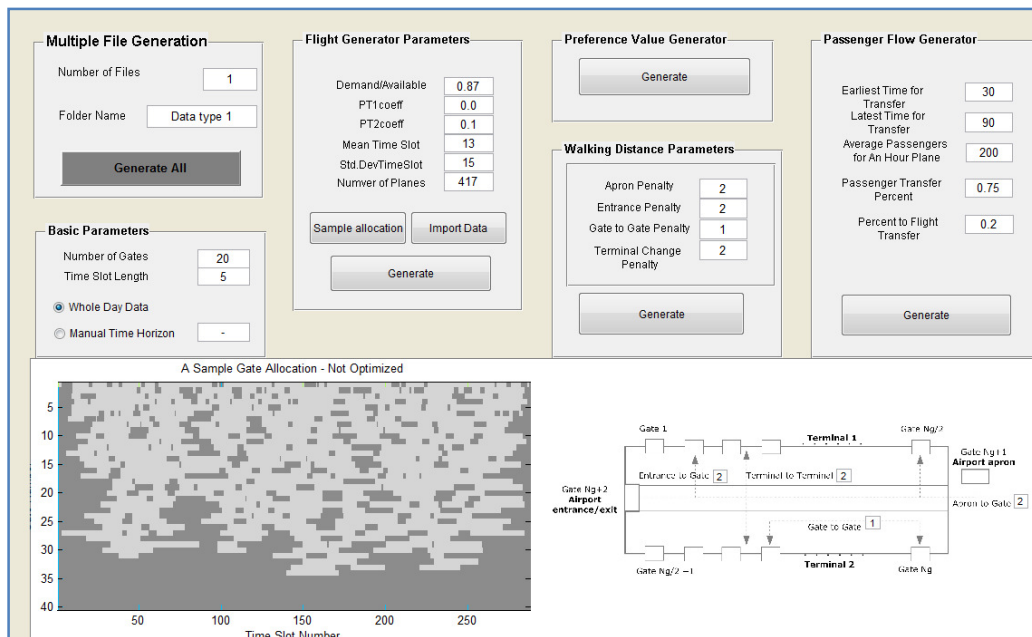
## 4.4 Simulation Results

### 4.4.1 Simulation results with artificially generated dataset

In this section, the performance results are provided to demonstrate the effect of SL-BBBC method over test data sets. Dedicated test data generator has been developed that considers the following parameters as inputs:

- I. the proportion in between total time slot demand and total available discrete time slots,  $d$
- II. prime time traffic factors,  $p1$  and  $p2$ ;
- III. mean staying time for a plane,  $m$ ;
- IV. standard deviation for staying times of all arranged flights,  $\sigma$ .

The user of this test data generator can define at most two prime time (high gate demand time interval during a day). When a flight is generated by the adjusted parameters, it will be assigned to each prime time with a probability equal to the chosen factor of the corresponding prime time. Even if it is not assigned to prime time by this step, the flight can still be assigned to that region by coincidence. The test data generator makes up the whole flight list accordingly. The mean, standard deviation and prime time factors are directly used in daily flight list generation. However, the ratio of demanded slot/available slot (= demanded gate duration / available gate time) is used to find the necessary plane number and then this value is assigned with some uniform random number in the vicinity of 10%. This simple manipulation is done just for the diversity of the plane numbers for batch data file generation. **Figure 4.** shows the graphical user interface for the test data generation software. For a sample (not optimized) view for the selected parameters, one may use “Sample allocation” button. Then, “Generate” button produces data files derived by the selected parameters at the instant. The passenger flow generator, walking distance parameters and preference value generator models are used in chapter 6.



**Figure 4.9 :** Graphical user interface for problem instance generation.

Data generation steps for a single day is as follows,

STEP 1: Find total gate duration demanded,  $T_t$ , in terms of discretized time slot number, (4.2)

$$T_t = d * N_g * N_t \quad (4.2)$$

where  $N_g$  is the number of gates and  $N_t$  is the number of time slots in a whole day.

STEP 2: Find number of planes,  $N$ , to be generated, (4.3),

$$N = \text{round}(T_t / m + 10 * \text{rand}) \quad (4.3)$$

where  $\text{rand}$  is a uniform random number in the interval  $[-1, 1]$ .  $\text{round}$  function produces the nearest integer as the number of planes should be an integer value.

STEP 3: For all  $N$  planes, pick up an integer gate duration value from normal distribution with mean  $m$  and standard deviation  $\sigma$  that are defined by the user.

STEP 4: For all  $N$  planes, assign the plane to the corresponding prime time region with a probability chosen by the user.

STEP 5: For the planes not allocated to the prime time regions, randomly assign arrival indexes that are convenient with the gate time determined in step 3.

The number of files to be produced can also be changed. “Number of Gates” parameter is arranged to depict preferable first  $N_g$  gates. In this specific example, the parameters are selected as follows:

$$N_g = 15, N_{oa} = 25 \text{ and } N_s = 40.$$

Three different files representing three different characteristics for a flight schedule are generated:

- 1) **Moderate data set:** This data set structure is close to data set structures observed in Turkey Airports. There is a relatively high demand for certain time slots during the morning and evening. Average gate time is close to an hour and median value is nearly half an hour. Demanded gate time does not exceed available gates. That is to say, the ratio of demanded time slot / available time slot is less than one. However, there occur un-gated flights due to lack of perfect fitting of gate durations.
- 2) **High gate demand distributed uniformly:** There are considerably larger demand for the same number of gates with data set-1. The ratio of demanded time slot / available time slot is slightly larger than one and this causes

irrepressible ungated flights. There are no intended peaks on the gate demands throughout the day.

- 3) **High gate demand with demand peaks:** Demanded gate duration / available gate time is same with data set-2. In this data set, the mean staying time for the flights are quite decreased (represents the flight schedule for an airport having very crowded traffic and many connection flights) and two regions of demand peak are defined, one of them hosting the 35% of the total flights.

Every experiment is carried for 100 times. The results are analyzed to yield mean, median, standard deviation, maximum and maximum of deviations and some of these are reported whenever appropriate.

Each stand at the airport has full vacancy at start. The whole day is divided into 5 minutes time intervals summing up  $24 * 60 / 5 = 288$  time slots. When scoring an assignment list, first  $N_g = 15$  stands are concerned. Each assigned time slot at the first  $N_g$  stands equally contributes to the scoring. For example, a plane arriving at 08:00am and departing at 11:00am stays for 36 time slots and if the plane can be assigned to one of the score contributing stands, the overall score for the assignment list will increase by 36.

**Table 4.1** reports the average results over 30-days. Note that the SL-BBBC algorithm is only allowed to run for 2500 fitness evaluations taking less than 1 minute in Intel Core 2 Duo Processor. Though three different approaches for SL-BBBC implementation have been tried through simulations, only the last one coded as SL-BBBC version-c is reported since it yielded the most successful results. **Figure 4.**, **Figure 4.** and **Figure 4.** show the cost scores of the three algorithms with respect to days.

**Table 4.1 :** Mean cost values for synthetic dataset (each consists of 30 days data).

| Method        | Moderate data set | High gate demand distributed uniformly | High gate demand having demand peaks |
|---------------|-------------------|--|--------------------------------------|
| Greedy Method | 2847.23           | 3004.50                                | 2913.77                              |
| GTMA          | 3440.33           | 3715.56                                | 3285.63                              |
| SL-BBBC       | 3483.67           | 3760.08                                | 3308.20                              |

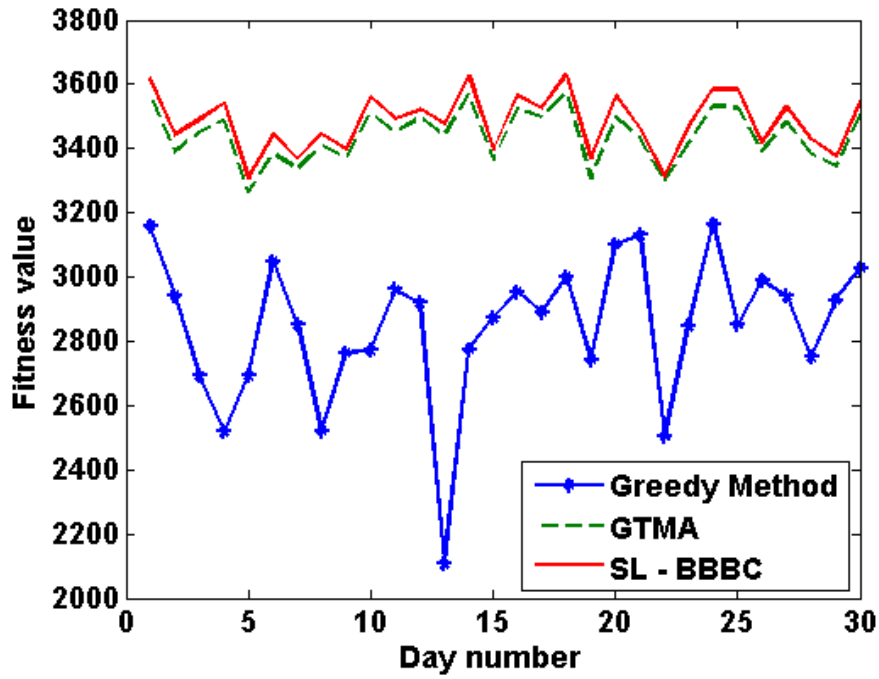


Figure 4.10 : Comparison of the three algorithms for moderate data set.

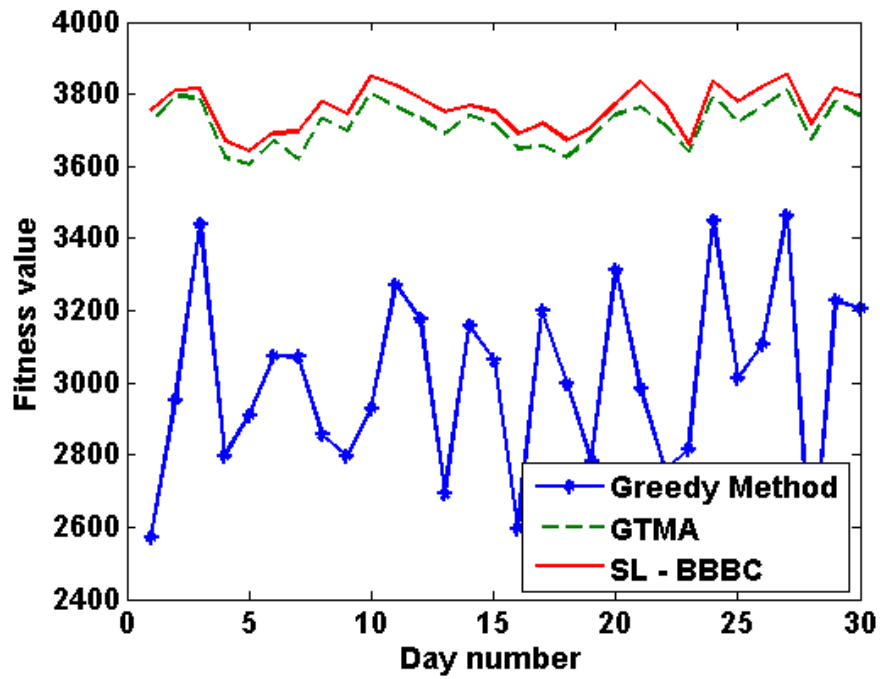
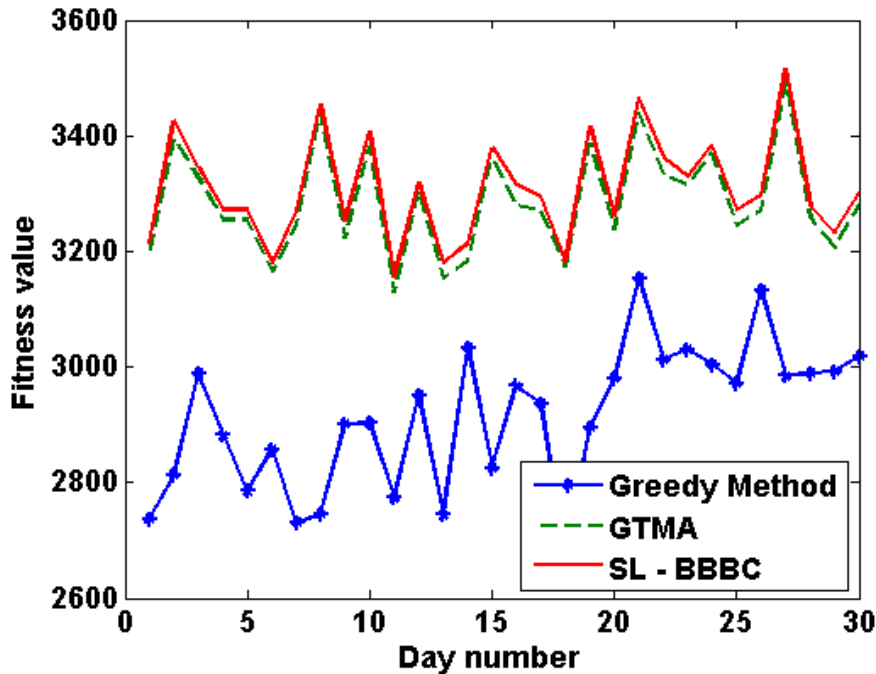


Figure 4.11 : Comparison of the three algorithms for high gate demand distributed uniformly data set.



**Figure 4.12 :** Comparison of the three algorithms for high gate demand with demand peaks data set.

#### 4.4.2 Simulation results with actual field data

In this part, the experiments are performed on the data collected from the Atatürk Airport in İstanbul. Data collected are for 31 days of month January 2009 and represent an average of 300 planes per day. The problem at hand is, again, squeezing maximum planes to the gates.

All the testing procedure and analyzing parameters are the same with tests performed in section 4.4.1.

**Table 4.2** reports the cost values of random ordering, in which the planes are ordered randomly (average on 100 random ordering for each day results are averaged over whole days); greedy method, GTMA, Random re-ordering over GTMA, where heuristic method's outputs are randomly interchanged for the same iteration number as SL-BBBC; and finally SL-BBBC method.

Greedy method performs even worse than random ordering average in this data set. Besides, even if one starts from a good initial point then it is observed that interchanging the plane orders in a totally random manner makes not much difference in performance; whereas, being a systematic method, applying SL-BBBC

algorithm with good initial conditions is still able to create respectful difference in performance. Note that the average cost score of random ordered allocations is somewhere near 2866 and GTMA heuristic improves this score by 16 percent. The stochastic neighborhood search method further improves the results by 9.6 percent of the previous improvement. That is to say, SL-BBBC algorithm starts from a quite acceptable solution and further improves the solution; on the other hand, if it had been started from a random solution candidate (that is a random ordering of the planes) the improvement would have been much more in the expense of process time. The algorithm is optimized both in terms of objective function value and process time by using an initial solution generated by a deterministic heuristic method.

**Table 4.2** : Mean cost values for 31 days.

| Method             | Real world data set |
|--------------------|---------------------|
| Random ordering    | 2866.71             |
| Greedy method      | 2761.52             |
| GTMA               | 3327.39             |
| Random re-ordering | 3333.00             |
| SL-BBBC            | 3371.53             |

The annual profit (company confidential) obtained by this final improvement justifies the importance of the new algorithm. **Figure 4.** clearly shows the improvement gained in using the SL-BBBC method in daily basis.

Since the total run time for the SL-BBBC algorithm is less than one minute it allows quick restructuring of the assignment table. That is one of the most powerful aspects of the algorithm for the practical applications. The method is compatible with any cost function evaluation but the algorithm speed heavily depends on cost function process time. Moreover, if the algorithm were allowed to evaluate more candidate fitness values, the objective function value scores could have been further improved.

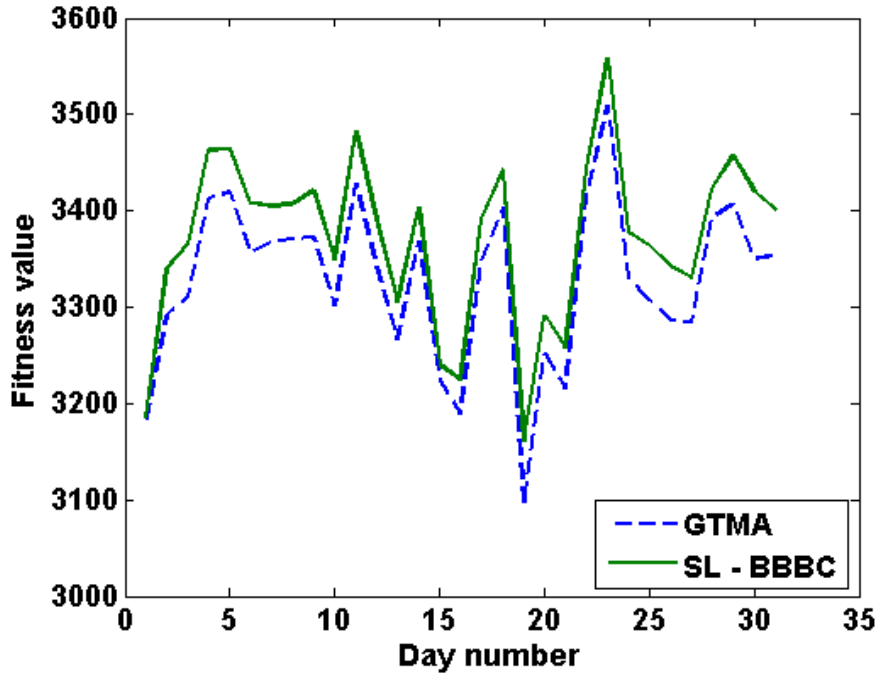


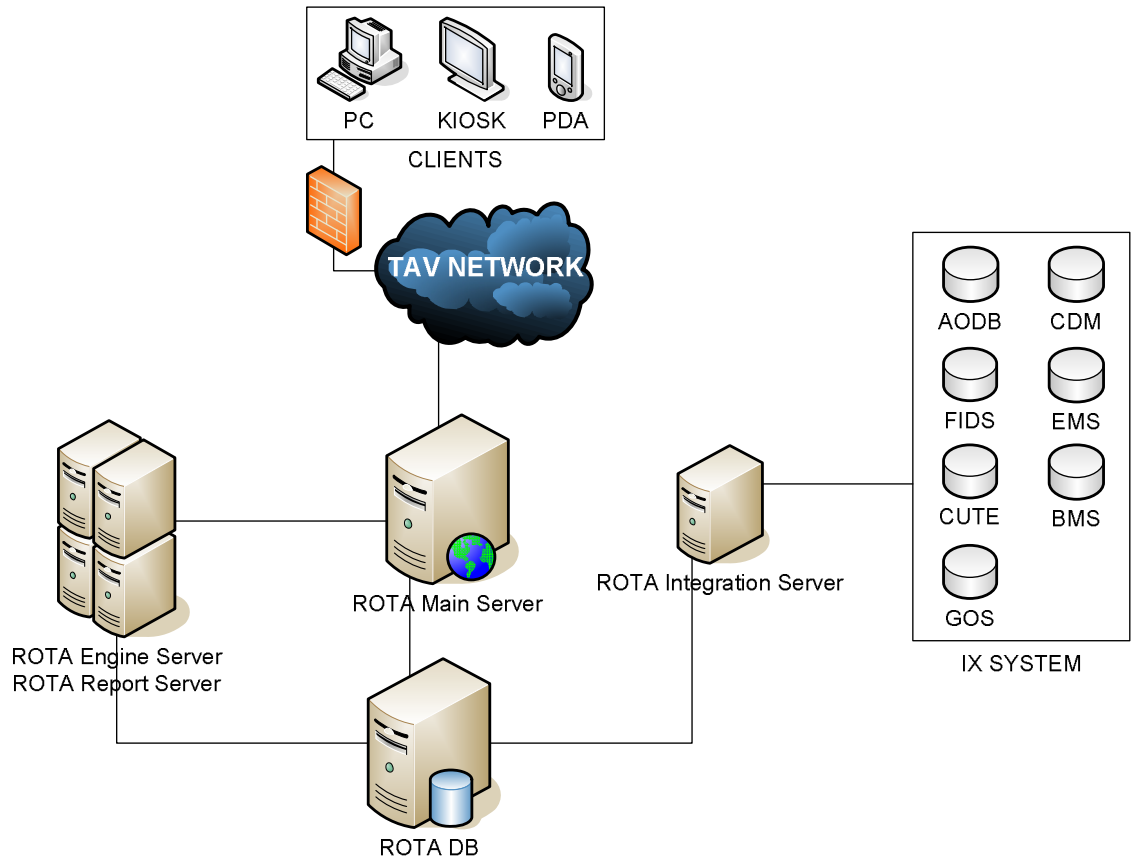
Figure 4.13 : Comparison of GTMA and SL-BBBC in a real world data set.

#### 4.5 Application at Atatürk Airport of İstanbul

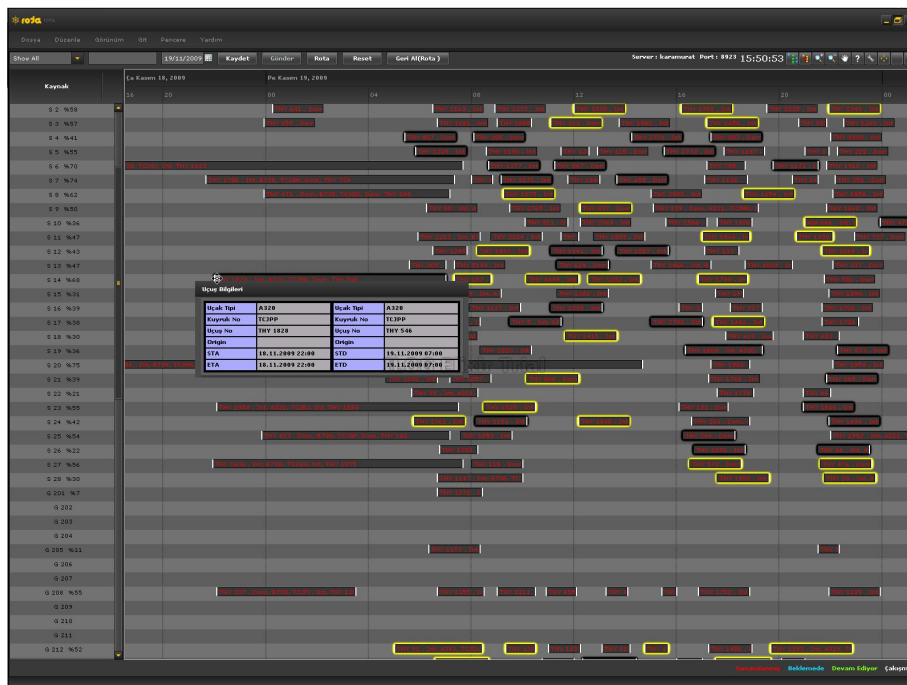
The proposed algorithm is used on a real world application to work both as an off-line and on-line gate allocation module in one of the most frequented airports of Europe, İstanbul Atatürk Airport. The software developed is a resource management system having the architecture given in **Figure 4.13**. The gate assignment automation is implemented on ROTA Engine Server. The detailed explanations for the other components are beyond the scope of this work and deliberately omitted here.

The Resource Management System (RMS) can be used as a web-based or desktop application. Thus, a variety of users with different devices throughout the airport can utilize the system in a collaborative manner. The user interface, Dashboard includes touch screen capability to maximize usability and control. Dashboard is designed mainly for maximizing monitoring capabilities according to the needs of control centre staff. **Figure 4.**, **Figure 4.** and **Figure 4.** gives some example screenshots from the Dashboard.

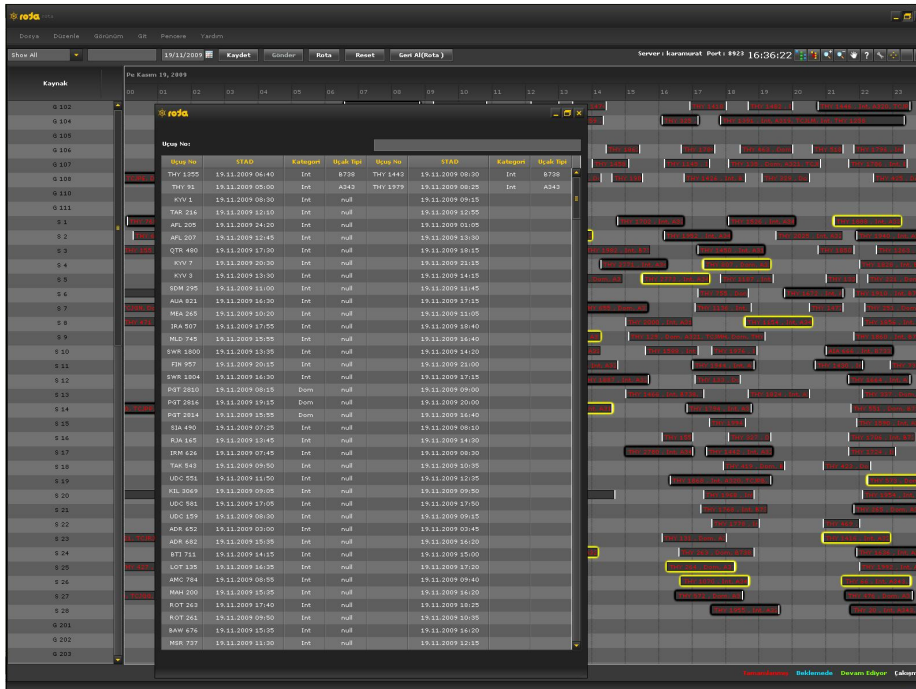




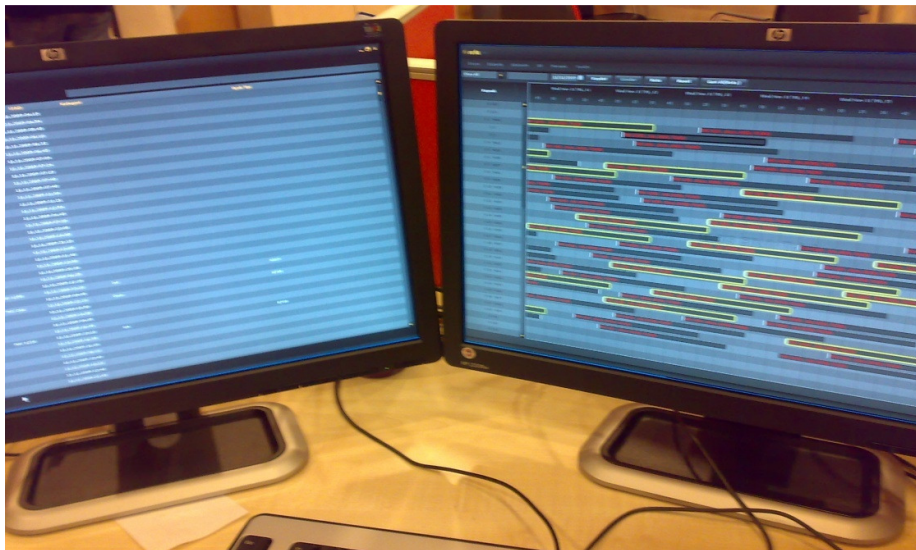
**Figure 4.14 :** The architecture for the resource management system.



**Figure 4.15 :** An example gate allocation screen when flight information window is open.



**Figure 4.16 :** An example gate allocation screen when manual edit window is open.



**Figure 4.17 :** Operator display for gate allocating (taken from Atatürk Airport with courtesy of TAV Bilişim A.S.)

The operator can display the statistics of assignment as well as the basic information about a particular flight. The flight list and the assignment list; the inputs and outputs of the system are displayed concurrently.

The core module of RMS is the Rule & Optimization Engine. The rule engine enables authorized personnel to model the constraints and relationships both for

resources and tasks through all their possible attributes according to market considerations and airline/agent preferences. These task-oriented rules can be based on the following groups and extended according to other classes and attributes:

- I. Airport-based
- II. Terminal-based
- III. Airline-based
- IV. Based on Registration Number
- V. Other (i.e. recurring assignment rules, etc.)

Predictably, because planning staff will utilize so many of the above constraints, it is inevitable that some of them will overlap. Although, in some cases, this can be a preferred result in terms of operational workload, some unexpected and undesired results may occur under normal circumstances. The rule engine module resolves this issue with a scoring mechanism and the overlapped constraints can be managed using this functionality. In addition to scoring and constraints, there is another functional parameter that can be accessed in modeling constraints called Soft Constraints (Preferences). These constraints refer to the rules, which can be violated by the Optimization Engine under some circumstances in order to meet functional objectives. At any given time during an operation (and in planning), if there is a shortage of resources, the system proactively generates automatic conflict messages with pre-defined solutions. It would be appropriate to point out at this stage that the SL-BBBC optimization algorithm operates independently from constraint generation.

Optimization Engine sub-module provides optimization of daily tasks based on pre-defined flight lists. The main idea in developing the optimization engine is to serve the priorities set by airport operators according to their management policies and preferences. Some of the common objective functions are given below:

- I. Revenue maximization through the optimization of gate and stand assignments.
- II. Maximizing the utilization of gate capacity.
- III. Maximizing airport capacity.
- IV. Enhancing overall service quality (punctual departures, cost competitiveness and reliability).

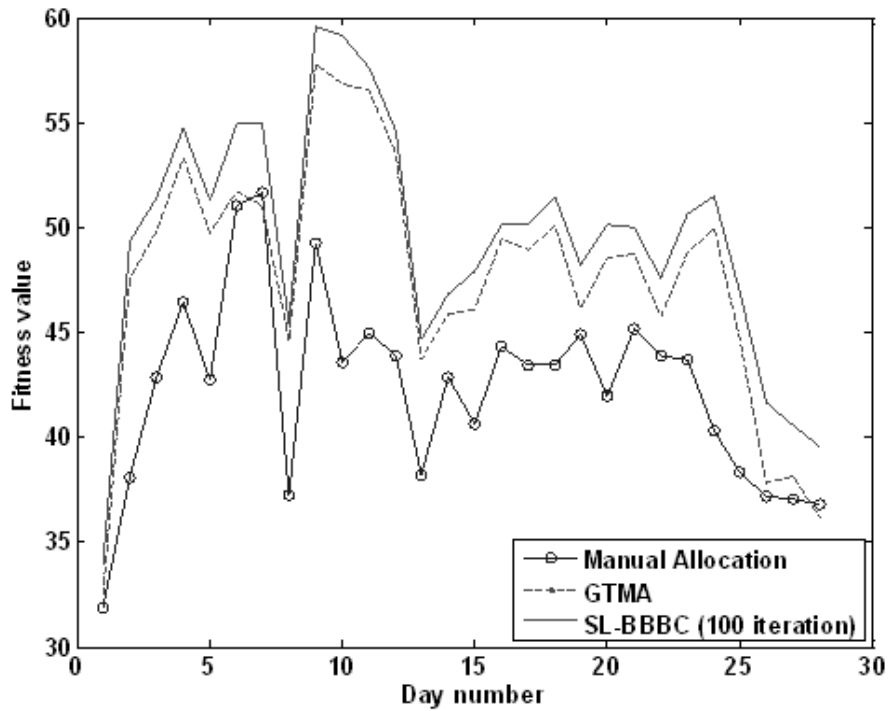
- V. Minimizing the walking distance for departing and arriving passengers to provide smooth passenger flow.

These objectives can be evaluated together according to their priority levels simply by defining the cost function as a weighted sum of the all listed above. It will again be appropriate to underline that SL-BBBC algorithm used here does not need to know this fitness function but just needs the score to proceed.

The performance results for the resource management system are given in **Table 4.3** and **Figure 4.**. In **Table 4.3**, the data collected for the month February of 2010 (28 days) are used and the scores are reported as the average for the month. The GTMA heuristic improves the cost by 12.49% with respect to manual (random) allocations and applying SL-BBBC algorithm for 100 iterations (= 100 “leaps”) improves the results by 22.68% of the previous improvement. For 1000 iterations, this improvement value is 34.59%. Fig.xxx gives the results in daily basis. The results for SL-BBBC algorithm for 100 iterations are omitted in order to decrease figure complexity.

**Table 4.3** : Mean cost values for the data collected from resource management system of Atatürk Airport for month February 2010.

| Method                     | Cost Score |
|----------------------------|------------|
| Manual allocations         | 2866.71    |
| GTMA                       | 3327.39    |
| SL-BBBC (100 evaluations)  | 3333.00    |
| SL-BBBC (1000 evaluations) | 3371.53    |



**Figure 4.18 :** Comparison of the algorithms running on Atatürk Airport. Data are the 28 days of February 2010.

#### 4.6 Conclusion

In this chapter, the airport gate assignment problem is considered as to maximize the total gate duration of the flights assigned to the gates. Then, the airport gate assignment problem turns out to be maximizing the total sojourn in the first  $N_g$  gates. A new stochastic approach has been introduced to the problem utilizing a problem specific modification of Big Bang-Big Crunch optimization algorithm, namely Single Leap-Big Bang Big Crunch (SL-BBBC). The key feature of this problem specific evolutionary optimization algorithm, that is also the one of the main contributions of this Chapter, is to interchange the queue order of the planes (=flights) to be assigned rather than interchanging the positions of the  $N$  planes that are already assigned. Therefore, the algorithm steps do not interact with the assignment strategy and they just exchange the order of plane handling by the determined strategy. This modularity of SL-BBBC makes it compatible with any assignment logic. This hybridized approach is shown on a simple yet effective heuristic algorithm, which is abbreviated as GTMA. Starting from a good initial

solution obtained by this heuristic and then using the newly proposed stochastic method is rather effective in terms of process time and the method proposed can be used in all practical applications.

The results obtained through simulation examples and experiments with real world data show the effectiveness of the allocation strategy. The modularity of the plane ordering logic provides great flexibility to work with any constraint-processing engine. Moreover, this new algorithm does not require any objective score calculation and does not have to know details on constraints or cost calculation. These facts are tried to be illustrated at the final section dedicated to the application study done on the biggest airport of Turkey.

## 5. INTRODUCTION TO MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

### 5.1 Multi-Objective Problem (MOP) Definitions and Basic Concepts

The multi-objective optimization problem (MOP) is defined by Osyczka (1985) as follows:

a vector of decision variables which satisfies constraints and optimizes a vector function whose elements represent the objective functions. These functions form a mathematical description of performance criteria, which are usually in conflict with each other. Hence, the term “optimize” means finding such a solution which would give the values of all the objective functions acceptable to the decision maker.

A general MOP is formally defined as minimizing (or maximizing) **(5.1)**

$$F(x) = (f_1(x), \dots, f_k(x)) \quad (5.1)$$

subject to

$$g_i(x) \leq 0, i = \{1, \dots, m\} \quad \text{and} \quad h_j(x) = 0, j = \{1, \dots, p\}$$

where  $f_i(x)$  is the objective function,  $m$  is the number of inequality constraints and  $p$  is the number of equality constraints. An MOP solution minimizes (or maximizes) the components of a vector  $\mathbf{x} = (x_1, \dots, x_n)$ .

Pareto Optimality: For a given MOP, pareto optimal set ( $P^*$ ) is defined as,

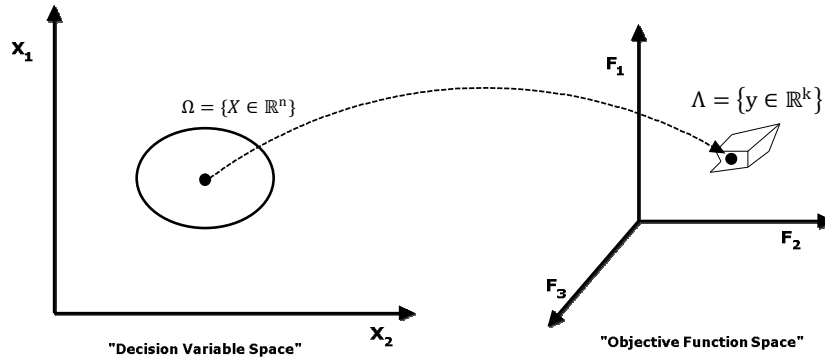
$$P^* := \{x \in \Omega \mid \neg \exists x' \in \Omega \text{ s.t. } F(x') \prec F(x)\}$$

The solutions in the Pareto optimal set are defined as non-inferior, admissible or efficient solutions. Corresponding genotypes are the nondominated vectors.

Pareto Front: For a given MOP and Pareto optimal set ( $P^*$ ), the pareto front ( $PF^*$ ) is defined as,

$$PF^* := \{u = F(x) \mid x \in P^*\}$$

Pareto Optimal Set is defined on genotype space whereas the Pareto front is the mapping on the phenotype space (**Figure 5.1**).



**Figure 5.1** : Mapping between genotype and phenotype space.

*Decision Making:* Pareto optimal solutions are those which when evaluated, produce vectors whose performance in one dimension cannot be improved without adversely affecting another. The global minimum or the single solution for a multi-objective problem can be obtained by selecting the best compromise solution in the Pareto optimal set. Selecting this single solution is the process of decision-making.

## 5.2 Classification of MOP Solution Techniques

A commonly accepted classification is based on the interaction between optimization and decision tradeoffs:

A priori preference articulation (make decisions before search): this group of techniques includes those approaches that assume either a certain desired achievable goal or a certain pre-ordering of the objectives can be performed by the decision maker (DM) prior to the search. The most common methods reported in the literature are listed as follows:

- I. Global Criterion Method
- II. Goal Programming
- III. Goal Attainment Method
- IV. Lexicographic Method
- V. Min-Max Optimization



- VI. Multi-attribute Utility Theory
- VII. Surrogate Worth Trade-Off
- VIII. ELECTRE- I
- IX. ELECTRE-II
- X. PROMETHEE

A posteriori preference articulation (search before making decisions): These techniques do not require prior preference information from the DM. These techniques do not require prior preference information from the DM. Some of the techniques included in this category are among the oldest multi-objective optimization approaches proposed:

- I. Linear Combination of Weights,
- II.  $\epsilon$ -Constraint Method.

Progressive Preference Articulation (integrate search and decision making): These techniques operate in 3 steps (Cohon and Marks, 1975):

STEP 1: find a nondominated solution,

STEP 2: get the reaction of the DM regarding this nondominated solution, and modify the preferences of the objectives accordingly

STEP 3: repeat the two previous steps until the DM is satisfied or no further improvement is possible.

General progressive preference articulation methods are,

- I. Probabilistic Trade-Off Development Method,
- II. STEP Method and
- III. Sequential Multi-objective Problem Solving Method.

### **5.3 Basic Concepts on Multi-Objective Evolutionary Algorithms (MOEAs)**

In many occasions, the problem domain is either too complex to be mathematically formulate or finding pareto optimal set through classical methods can be tremendously difficult. These types of problems can be effectively handled utilizing evolutionary routines. The basic algorithm design concept is to use Pareto-based

fitness assignment to identify nondominated vectors from a MOEA's current population. A generic MOEA steps can be summarized as follows (Coello Coello et al, 2007):

STEP 0: Define the MOP:

- determine the mathematical form of objective vector
- determine chromosome representation
- define constraints (dynamic, static, linear, nonlinear, etc.)
- integrate the model into a specific MOEA algorithmic search process.

STEP 1: The MOEA generates  $PF_{known}$  (hard part):

Determine the nondominated sets, generation to generation, via populations.

Converge "close" to the true computational Pareto front,  $PF_{true}$ .

STEP 2: The MOEA attempts to generate a uniform distribution across the known Pareto front,  $PF_{known}$ , at the end of each generation.

STEP 3: Select several of the optimal points on the pareto front,  $PF_{known}$ , for decision maker (DM) consideration.

STEP 4: Determine the associated pareto optimal set,  $P_{known}$ ; implement decision variable values (i.e., approximation of the Pareto optimal set) as selected by the DM.

STEP 5: Visualize algorithm processing and results as appropriate for improving MOEA performance (i.e., efficiency and effectiveness).

Through these steps, a MOEA serves for the following goals:

- I. to preserve nondominated points (elitism vs. non-elitism) with  $PF_{current} \rightarrow PF_{known}$
- II. to progress or guide  $PF_{known}$  towards  $PF_{true}$
- III. to generate and maintain diversity of points on the  $PF$ , ( $PF_{known}$  (phenotype) and/or Pareto optimal solutions  $P_{known}$ )
- IV. Provide the decision maker (DM) with a limited number of  $PF_{known}$  points.

#### 5.4 Pareto Based MOEA Concepts

A solution strategy for a multi-objective problem introduces three main issues (over its single objective counterpart). To extract a population of nondominated solutions,

dominance based ranking, diversity preservation and secondary population management concepts are vital.

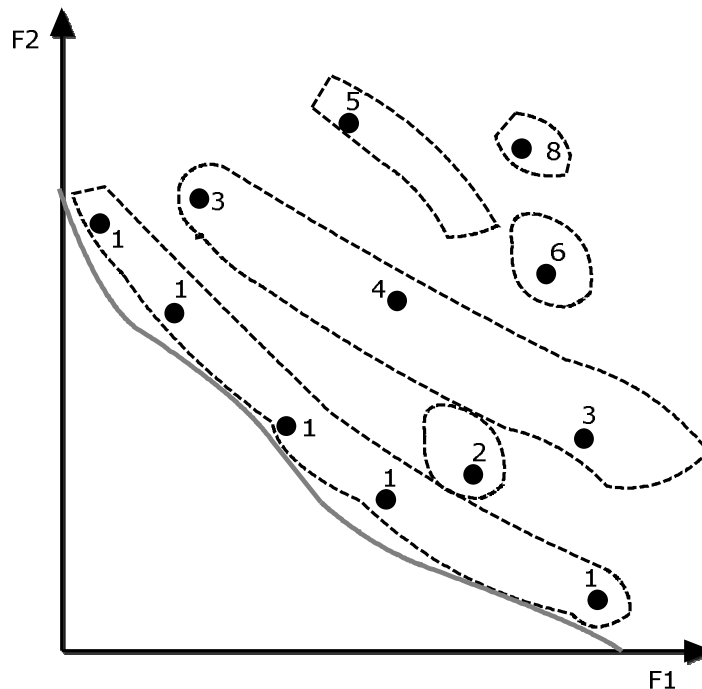
#### 5.4.1 Dominance-based ranking or fitness assignment

Dominance operator is binary and has two possible results: either one operand dominates or they do not dominate each other. Besides, dominance operator is transitive, that is, propositions “A dominates B” and “B dominates C” requires “A dominates C”.

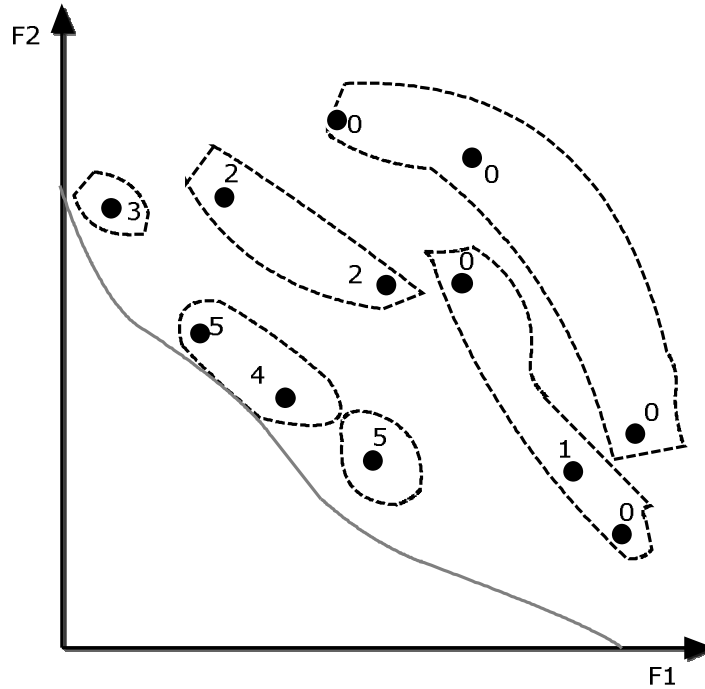
Regarding the selection and generation of the *PF*, an ordering method is needed based on dominance concept. There are three commonly accepted methods on dominance-based ranking:

- I. dominance rank: How many individuals is an individual dominated by (plus 1)? (See **Figure 5.2**)
- II. dominance count: How many individuals does an individual dominate? (**Figure 5.3**)
- III. dominance depth: At which front is an individual located? Sort.

Given a particular problem domain, selecting any of the dominance based ranking method, varies the performance (efficiency and effectiveness) considerably.



**Figure 5.2 :** Dominance rank.



**Figure 5.3 :** Dominance count.

### 5.4.2 Diversity preservation

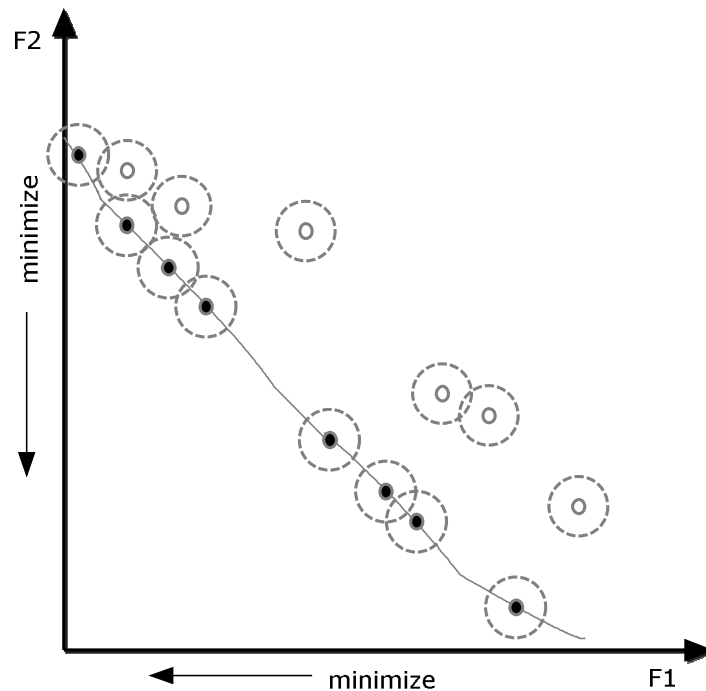
Another important issue in designing a MOEA is diversity preservation. Finding nondominated members is expensive in terms of calculation, so it is important to select appropriate members to the mating pool as much as possible. To achieve this, diversity in the population should be preserved and all the search space must be scanned with nearly equal probability. At the end of the day, the ultimate goal is to provide a diverse set of  $PF_{known}$  or  $P_{known}$  points (having a uniform distribution across the known  $PF$ ) to the DM. The diversity preservation methods can be investigated in five categories:

- 1) **Weight Vector Approach:** A vector set in fitness/objective space is used to attempt to diversify points of the Pareto front surface. By changing the weights, different directions are defined, in order to bias the search, and to move solutions away from its neighbors.
- 2) **Fitness Sharing/Niching Approach:** In most general case of fitness sharing all the members within a certain radius  $\sigma_{share}$  is penalized. This radius is frequently called as niche radius. The definition of the niche radius is critical for algorithm success.

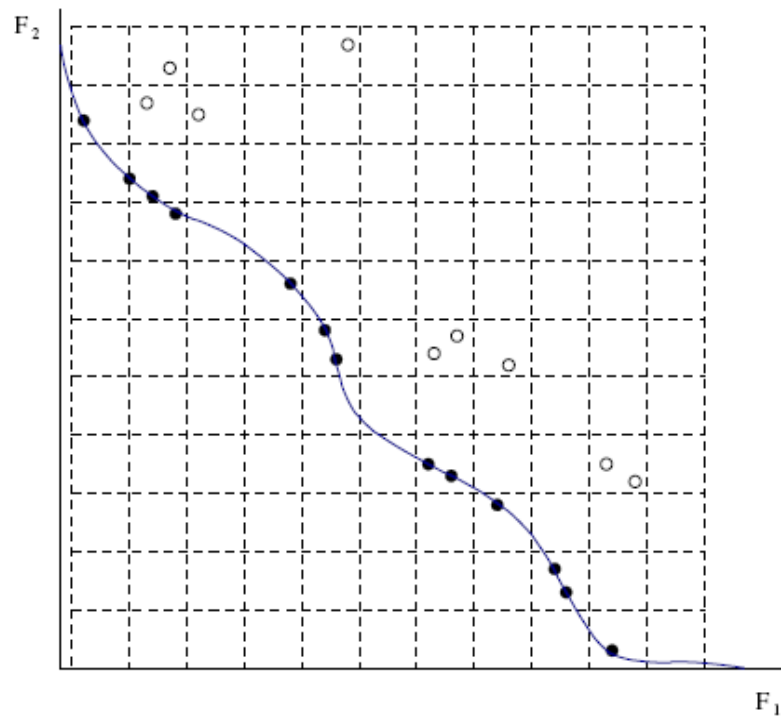
In order to apply a fitness sharing function, it is necessary to measure distances. Such distances can be measured in genotype or phenotype space. Illustrations for the fitness sharing approach is given in **Figure 5.4** and **Figure 5.5**.

Other most common fitness sharing approaches are as follows:

- Kernel approach: The density estimator is based on the sum of distance (vector) measured either in genotypic or in phenotypic space.
- Nearest neighbor approach: The density estimator is based on the volume of the hyper-rectangle defined by the nearest neighbors.
- Histogram approach: The density estimator is based on the number of solutions that lie within the same hyper-box.



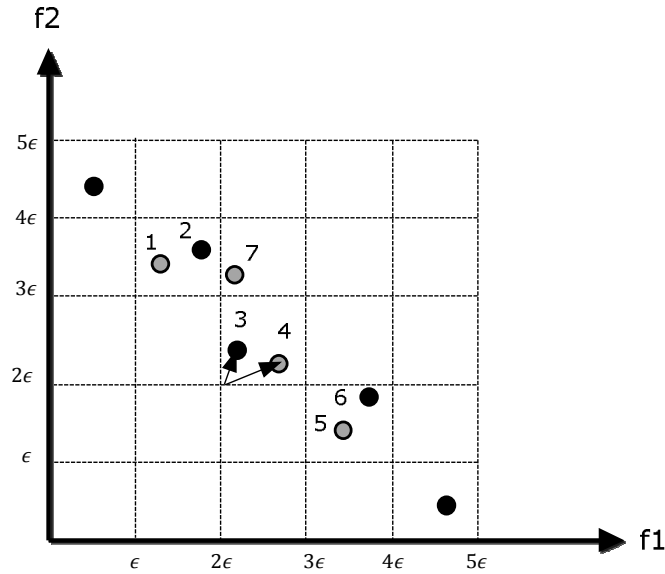
**Figure 5.4** : An illustration of fitness sharing.



**Figure 5.5 :** Another implementation of fitness sharing: Niching by gridding (The figure is taken from Coello coello et al. (2007)).

- 3) Crowding/Clustering: The main idea is selecting the surviving solutions according to region crowdedness metric measured in objective function space. The approach is similar with fitness sharing but more efficient in terms of computation complexity. In clustering, many points can be induced to one representative point. Both approaches provide the elimination of excessive members before dominance degrees are calculated.
- 4) Restricted Mating: In this case, diversity is preserved through the avoidance of certain recombinations. A parameter ( $\sigma_{mate}$ ) is defined for the minimum distance that must separate two individuals so that they can mate.
- 5) Relaxed Dominance: Key point in relaxed domination forms is to use a certain solution  $x$  even though it is worse than some solution  $y$  in regards to a particular objective (value comparison in objective function space). This relaxation may be compensated by an improvement in other objectives. As an example in **Figure 5.6**, if there are more than one nondominated point (since there are two dimensions, every point can be better in only one dimension in comparison to the other point) in the same grid, the one improving the most can be taken as nondominated pruning the others. So, in this minimization

problem point 3 is selected over 4 in the nondominated set. Note that 2 is dominated by 1 and 6 is dominated by 5 and they are out of the scope of relaxed dominance definition.



**Figure 5.6 :** A relaxed dominance form.

## 5.5 MOEA Population Structure

In parameter space, two population structures exist:

- 1)  $P_{known}$ : obtained nondominated solutions, updated periodically. (also called archival, external, secondary population) . It can be perceived as the multi-objective counterpart of the elitism concept.
- 2)  $P_{current}$ : main population evolving. Periodically some members or offspring can promote to the archival population. (Main evolution population is also called as primary or generational population)

There are continuing discussions on the management of the secondary population. The main question is “Actively involve  $P_{known}$  in evolution process or not?” The addressed issues are:

- I. Continuous addition and culling (choosing the addition and culling criteria)
- II. Update period selection
- III. Clustering or culling in case of size overflow
- IV. Homogenizing population distribution or remedy holes in the distribution

## 5.6 Baseline Algorithms

There are many evolutionary multi-objective optimization algorithms reported in the literature. Some of the most used versions include:

- I. Multi-Objective Genetic Algorithm (MOGA)
- II. Multi-Objective Genetic Algorithm-II (MOGA-II)
- III. Nondominated Sorting Genetic Algorithm (NSGA)
- IV. Nondominated Sorting Genetic Algorithm-II (NSGA-II)
- V. Niche-Pareto Genetic Algorithm (NPGA)
- VI. Niche-Pareto Genetic Algorithm-II (NPGA-II)
- VII. Pareto Archived Evolution Strategy (PAES)
- VIII. Strength Pareto Evolutionary Algorithm (SPEA)
- IX. Strength Pareto Evolutionary Algorithm-II (SPEA-II)

In this chapter, only NSGA-II, PAES and SPEA-II are briefly reviewed since they are used in the most common sense.

### 5.6.1 Nondominated sorting genetic algorithm-II

There were three main disadvantages of MOEAs up to 2000s (Deb et al, 2000, 2002):

- I.  $O(MN^3)$  computational complexity (where M is the number of objectives and N is the population size)
- II. Non-elitism
- III. Need for specifying a sharing parameter.

NSGA-II is proposed to overcome all these disadvantages (Deb et al, 2000). To improve worst case computational complexity, “Fast Non-Dominated Sorting Approach” is implemented:

STEP 1: Calculate (for each individual)

- I. Number of solutions that dominate  $p$
- II. Set  $S_p$  of solutions that the solution  $p$  dominates

STEP 2: Take members having domination count 0 to the first front. Then visit its set and decrease the dominance count of the members by one. If any set becomes 0; they



constitute the second front. The process terminates once all fronts are identified. (dominance depth)

For each solution in the second or higher level of nondomination, the domination count can be at most  $(N - 1)$ . Thus, each solution  $p$  will be visited at most  $(N - 1)$  times before its domination count becomes zero. At this point, the solution is assigned a nondomination level and will never be visited again. Since there are at most  $(N - 1)$  such solutions, the total complexity is  $O(N^2)$ . Thus, the overall complexity of the procedure is  $O(MN^2)$ .

The performance of the sharing function method in maintaining a spread of solutions depends largely on the chosen  $\sigma_{share}$  value. Since each solution must be compared with all other solutions in the population, the overall complexity of the sharing function approach is  $O(N^2)$ . However, NSGA-II algorithm introduces fast crowding distance estimation procedure. This procedure can be summarized as follows:

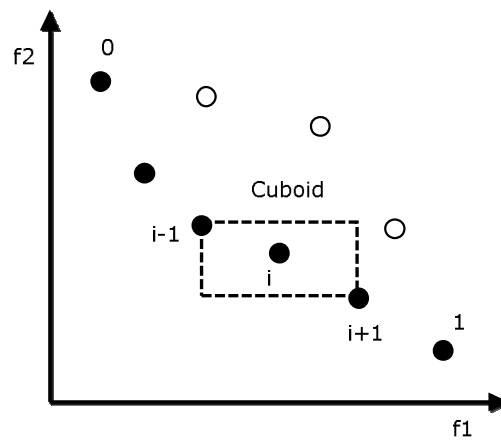
STEP 1: Sort population according to each objective.

STEP 2: Boundary solutions are assigned an infinite distance value.

STEP 3: For other solutions assign a distance value equal to the absolute normalized difference in the function values of two adjacent solutions.

STEP 4: Perform this calculation for all objectives.

STEP 5: The overall crowding-distance value is calculated as the sum of individual distance values corresponding to each objective (objectives are normalized) (**Figure 5.7**).



**Figure 5.7** : Crowding distances.

Sorting algorithm governs the complexity of this procedure. Since  $M$  independent sortings of at most  $N$  solutions (when all population members are in one front) are involved, the above algorithm has  $O(MN \log N)$  computational complexity.

- I. In NSGA-II, a simple crowded comparison operator is used to ensure uniform spread over the pareto front by guiding selection process. Every member has two attributes: Nondomination rank ( $i_{rank}$ )
- II. Crowding distance ( $i_{distance}$ , Average length of the sides of the cuboid along the objectives in **Figure 5.7**).

Then the crowding comparison operator is  $i < j$  if  $i_{rank} < j_{rank} \parallel (i_{rank} = j_{rank} \ \&\& \ i_{distance} > j_{distance})$ , where  $<$  is the crowded-comparison operator.

The pseudocode for NSGA-II algorithm is given in **Table 5.1**.

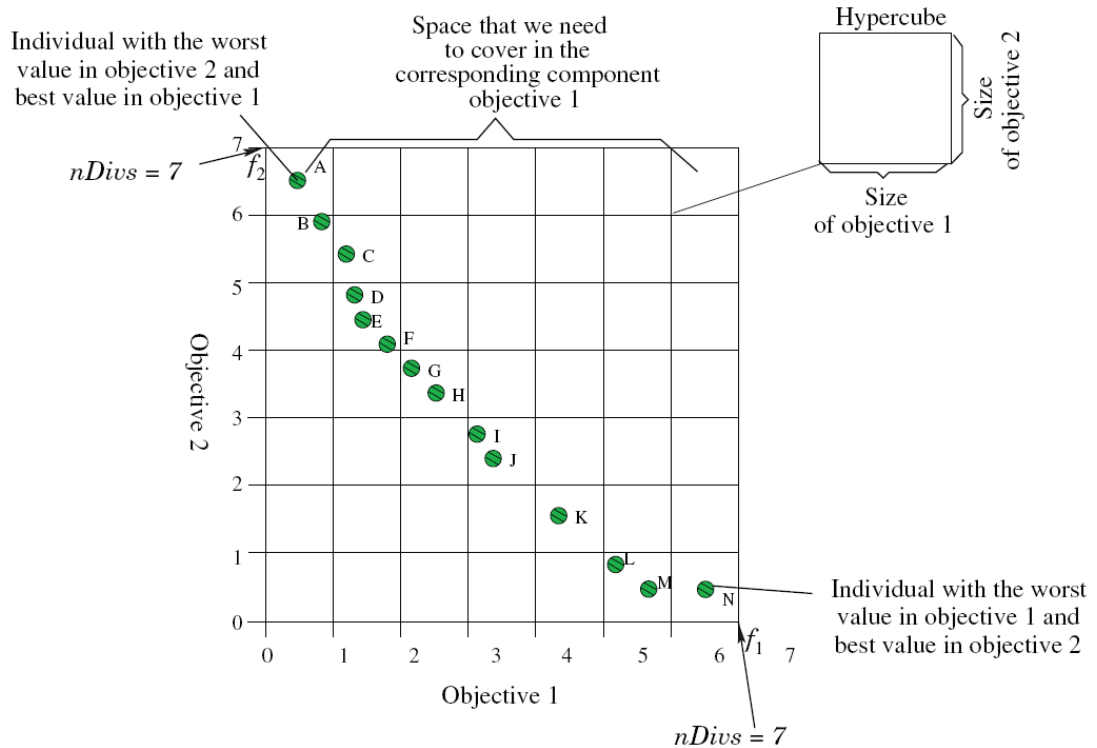
**Table 5.1** : Pseudocode for NSGA-II.

|  |
|--|
| <pre> Initialize Population P Evaluate Objective Values Assign Rank (level) Based on Pareto dominance - sort Generate Child Population of size N     Binary Tournament Selection     Recombination and Mutation for i = 1 to g do     for each Parent and Child in Population do (combined population~elitism)         Assign Rank (level) based on Pareto - sort         Generate sets of nondominated vectors along PF<sub>known</sub>         Loop (inside) by adding solutions to next generation starting from the first         front until N individuals found         Determine crowding distance between points on each front         Select points (elitist) on the lower front (with lower rank) and are outside a         crowding distance         Create next generation             Binary Tournament Selection             Recombination and Mutation </pre> |
|--|

### 5.6.2 Pareto archived evolution strategy

PAES algorithm (Knowles and Corne, 2000) is the multi-objective version of the evolution strategies. It introduces a recursive crowding procedure named as *adaptive gridding*. Each solution is placed in a certain grid location based on the values of its objectives (which are used as its coordinates or geographical location). A map of such grid is maintained, indicating the number of solutions that reside in each grid

location. Since the procedure is adaptive, no extra parameters are required (except for the number of divisions of the objective space). Furthermore, the procedure has a lower computational complexity than traditional niching methods (**Figure 5.8**).



**Figure 5.8 :** Adaptive Gridding algorithm (The figure is taken from Coello coello et al. (2007)).

Pseudo code for single individual case is given in **Table 5.2**. A historical archive (~secondary population) is used as a reference set against which each mutated individual is being compared. The pseudo code for archiving test is given in

Table 5.3.

**Table 5.2 :** Pseudocode for (1+1) PAES

|   |
|---|
| <pre> repeat   Initialize Single Population parent, <math>C</math>, and add to archive, <math>A</math>   Mutate <math>C</math> to produce child <math>M</math> and evaluate fitness   if <math>C</math> dominates <math>M</math>     discard <math>M</math>   else if <math>M</math> dominates <math>C</math>     replace <math>C</math> with <math>M</math>, and add <math>M</math> to archive   else if <math>M</math> is dominated by any member in the archive     discard <math>M</math>   else     apply test (<math>C, M, A</math>) to determine which becomes the new current solution     and whether to add <math>M</math> to <math>A</math> </pre> |
|---|

**Table 5.3 :** Pseudocode for archiving test in PAES

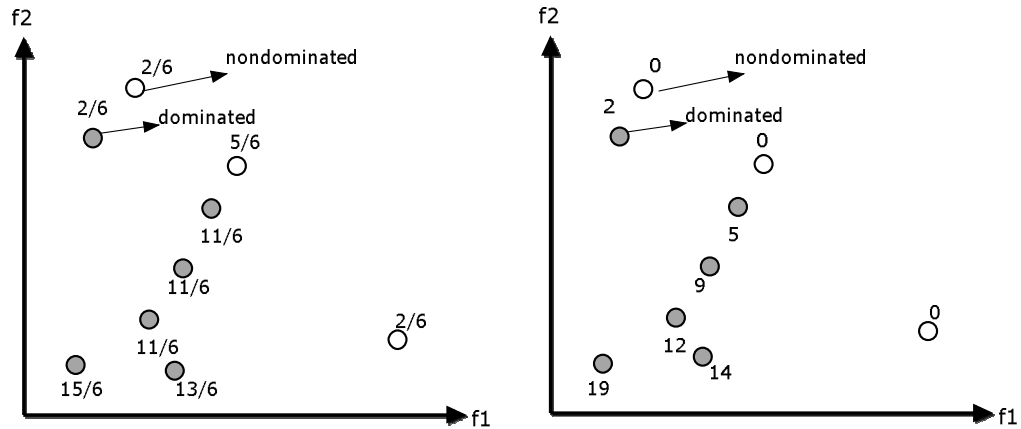
```
If the archive is not full
  Add  $M$  to the archive
  if  $M$  is in a less crowded region of the archive than  $C$ 
    Accept  $M$  as the new current solution
  else
    Maintain  $C$  as the current solution
else
  if  $M$  is a less crowded region of the archive than  $X$ , for some member  $X$  in the
  archive
    Add  $M$  to the archive and remove a member of the archive from the most
    crowded region
    If  $M$  is in a less crowded region of the archive than  $C$ 
      Accept  $M$  as the new current solution
    else
      Maintain  $C$  as the current solution
  else
    if( $M$  is in a less crowded region of the archive than  $C$ )
      Accept  $M$  as the new current solution
    else
      Maintain  $C$  as the current solution
```

### 5.6.3 Strength pareto evolutionary algorithm-II

The key idea of the strength pareto evolutionary algorithm is to assign degrees to dominating members in terms of strength measures. SPEA-II (Zitzler et al, 2001) takes forward to this approach and introduces a fine-grained fitness assignment strategy, a new density estimation technique and enhanced archive truncation method. The archive size is fixed, i.e., whenever the number of nondominated individuals is less than the predefined archive size, the archive is filled up by dominated individuals (with SPEA, the archive size may vary over time). In addition, the clustering technique, which is invoked when the nondominated front exceeds the archive limit, has been replaced by an alternative truncation method which has similar features but does not loose boundary points. Only members of the archive participate in the mating selection process.

Individuals that are dominated by the same archive members have identical fitness values. When the archive contains only a single individual, all population members have the same rank independent of whether they dominate each other or not.

Therefore, the selection pressure is decreased substantially and in this particular case, SPEA behaves like a random search algorithm. In SPEA-II, members are penalized with respect to domination depth (**Figure 5.9**).



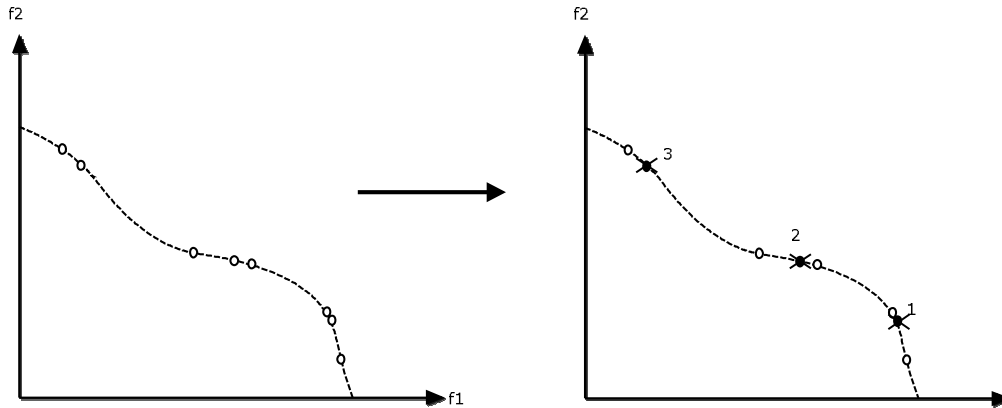
**Figure 5.9 :** Strength assignments of SPEA and SPEA-II.

Although the raw fitness assignment provides a sort of niching mechanism based on the concept of pareto dominance, it may fail when most individuals do not dominate each other. Therefore, additional density information is incorporated to discriminate between individuals having identical raw fitness values (**5.2**).

$$F(i) = R(i) + D(i) \quad (5.2)$$

where  $F(i)$  is the fitness for  $i^{th}$  member;  $R(i)$  is the dominance based ranking value and finally  $D(i)$  is the density measure for the  $i^{th}$  member.

Although the clustering technique used in SPEA is able to reduce the nondominated set without destroying its characteristics, it may lose outer solutions. However, these solutions should be kept in the archive in order to obtain a good spread of nondominated solutions. Therefore, a simple archive truncation method is adopted (**Figure 5.10**).



**Figure 5.10 :** Illustration of the archive truncation method used in SPEA2. On the right, a nondominated set is shown. On the left, it is depicted which solutions are removed in which order by the truncate operator.

Pseudo code for SPEA-II algorithm is given in **Table 5.4**.

**Table 5.4 :** Pseudocode for archiving test in SPEA-II

|  |
|--|
| <pre> Initialize Population P Create empty external set E for i=1 to g do     Compute fitness of each individual in P and E     Copy all individual evaluating to nondominated vectors P and E to E     Use the truncation operator to remove elements from E when the capacity of the     file has been extended     If the capacity of E has not been exceeded then use dominated individuals in P to     fill E     Perform binary tournament selection with replacement to fill the mating pool     Apply crossover and mutation to the mating pool </pre> |
|--|

## 5.7 MOEA Testing

The main goal of testing is usually to compare MOEA effectiveness over various chosen MOPs by measuring solution quality. The test functions used and the metrics differ quite a lot in comparison with the single objective counterpart. Every algorithm can maintain a group of nondominated individuals at the end of the run. Sometimes the result from one algorithm fully dominates the other, which is the simplest condition. However, generally, some results from one algorithm dominate some from another algorithm, and vice versa. Another reason for the special consideration on the performance evaluation is that one is interested in not only the convergence to  $PF_{true}$  but also the distribution of the individuals along  $PF_{true}$ .

Adequately evaluating convergence and distribution is still an open problem in the field of MOEAs (Yu and Mitsuo, 2010).

Using a test suite of any kind can be useful from a pedagogical perspective in comparing MOEAs, but in general, may be of little importance when solving real-world problems. Supporting this judgment, the no free lunch theorem (NFL) states “if problem domain knowledge is not incorporated into the algorithm domain, no formal assurances of an algorithm’s general robust effectiveness exist.” Still, a commonly accepted method is to use test functions first, then performing problem specific modifications. These test functions must have the following properties (Husband et al, 2006):

- I. The Pareto solution set should not reside at the edge of the feasible domain.
- II. The Pareto solution set should not reside in the center of the domain.
- III. Benchmark MOPs should have a scalable number of variables so that the designer and the analyzer can generate arbitrary dimensional MOPs.
- IV. Benchmark MOPs should have a scalable number of objectives.
- V. The variables of benchmark MOPs should have definition domains of different magnitudes. This characteristic tests the ability to change mutation strengths with different variables or the normalization ability of the algorithm.
- VI. The magnitudes of different objectives in  $PF_{actual}$  should be different.
- VII. The  $PF_{actual}$  of the problem can be expressed in explicit expression.

### 5.7.1 Basic test suites for multi-objective evolutionary algorithms

Most commonly used test suites in the literature are given here for complete referencing.

1. Van Veldhuizen summarized the multi-objective test problems before 1999 and selected seven of them as the benchmark (Van Veldhuizen, 1999).
2. ZDT: In 1999, Deb suggested a way to construct multi-objective test problems systematically (Deb, 1999). In Deb’s method, there is a function  $h$  to control the shape of  $PF_{true}$ , a function  $g$  to test the MOEAs’ ability to converge to  $PF_{true}$ , and a function  $f_1$  to test the MOEAs’ ability to distribute the individuals along  $PF_{true}$ . In

2000, Zitzler et al. used Deb's method to generate six benchmark MOPs (Zitzler et al, 2000).

3. DTLZ: In 2001, Deb et al. developed ZDT to nine scalable benchmark problems (Deb et al, 2001).

4. OKA: In 2004, Okabe et al. suggested another way to generate benchmark MOPs with an arbitrary Pareto optimal set shape and  $PF_{true}$  shape (Okabe et al, 2004). Apart from two examples to illustrate the effectiveness of the method, Okabe et al. also introduced a way to measure the convergence difficulty in OKA.

5. WFG: In 2005 and 2006, Husband et al. suggested a new scalable benchmark MOP suite with nine problems that contain and consider the characteristics and features discussed above (Husband et al, 2005, 2006).

6. In 2006, Iorio and Li pointed out that rotation might introduce difficulties for MOEAs and suggested four rotated benchmark MOP examples (Iorio and Li, 2006).

7. In 2006, Deb et al. addressed the importance of parameter dependencies for designing MOP benchmark problems and developed their ZDT and DTLZ through variable linkage (Deb et al, 2006)

8. In 2007 and 2009, the IEEE Congress on Evolutionary Computation held special sessions on multi-objective optimization and multi-objective optimization with constraints, respectively. The technical reports illustrate the corresponding benchmark problems (Huang et al, 2007, Zhang et al, 2008).

9. In 2009, Li and Zhang provided a new way of generating MOP benchmark problems with arbitrary prescribed  $PF_{true}$  shapes and gave nine examples (Li and Zhang, 2009).

## 5.8 Metrics of Performance

Comparing different optimization techniques experimentally always involves the notion of performance (Yu and Mitsuo, 2010). Performance is not only correlated with the convergence of the population; but also homogeneous distribution over the pareto front and the coverage:

- 1) Convergence: The distance of the resulting nondominated set to the Pareto-optimal front should be minimized.



- 2) Distribution: A good (~uniform) distribution of the solutions found is desirable (needs a distance metric).
- 3) Coverage: extent of the obtained nondominated front should be maximized.

The performance indices can be grouped in 7 main categories:

- I. Cardinality-based Performance Indices: If  $PF_{true}$  is known, the performance of the algorithm can be found by comparing the members in  $PF_{known}$
- II. Volume-based Performance Indices: The size of the space dominated by the final pareto set is used.
- III. Distance-based Performance Indices: If  $PF_{true}$  is known, the distance of the members in the found pareto set to the members in  $PF_{true}$  is used.
- IV. Attainment Surface-based Performance Indices
- V. Distribution Performance Indices: The homogeneity of the solution set is considered.
- VI. Spread Performance Indices: the spread (extent) of the pareto front is used.
- VII. Distribution and Spread Performance Indices: Both the spread and the distribution is considered.

In this thesis, for evaluating the performance of the MOEAs in chapter 6, following performance metrics will be used.

**Two set coverage,  $CS(S_1, S_2)$  (Zitzler, 1999):** is a binary metric that assigns dominance degrees to two sets of data.  $CS(S_1, S_2)$  is defined as the percent of the individuals in the second set,  $S_2$  who are weakly dominated by first set,  $S_1$ , (5.3). That is, the larger  $CS(S_1, S_2)$  is, the better  $S_1$  outperforms  $S_2$ .

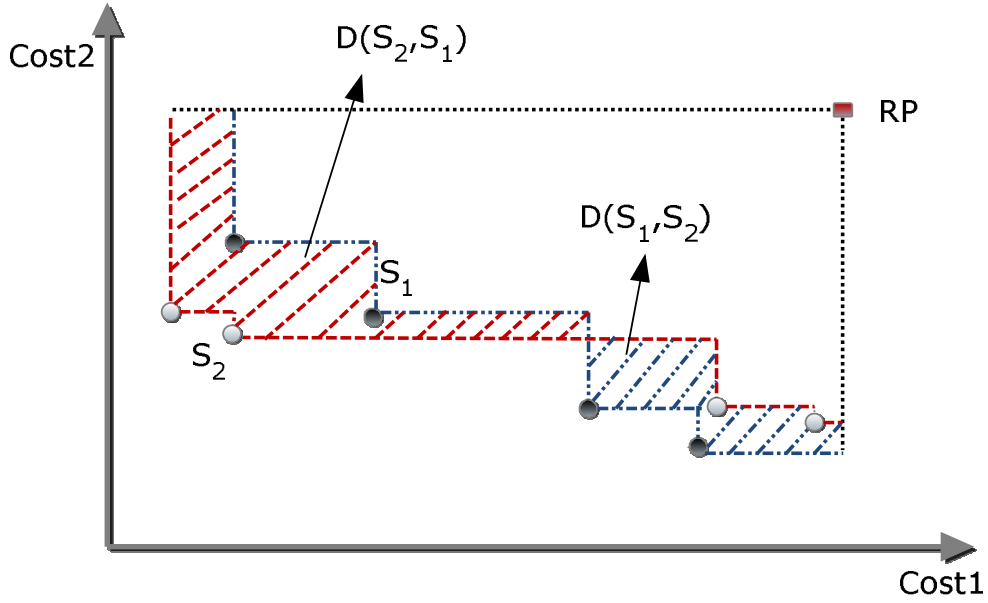
$$CS(S_1, S_2) = \frac{|[S_2 \in S_2] \exists s_1 \in S_1 : s_1 \preceq s_2|}{|S_2|} \quad (5.3)$$

**Hyper-volume (HV) (Zitzler et al, 2003):** is the size of the space dominated by a pareto front of solutions. In calculating HV, one needs to decide a reference point to compute dominated space size. This point is illustrated as *RP* point in **Figure 5.11**.

**Coverage Difference,  $D(S_1, S_2)$  (Zitzler, 1999):** is a binary implementation for hypervolume metric that emphasizes on the dominating areas of both sets. In **Figure 5.11**, areas corresponding to  $D(S_1, S_2)$  and  $D(S_2, S_1)$  are illustrated.

$$D(S_1, S_2) = HV(S_1 + S_2) - HV(S_2) \quad (5.4)$$

$$D(S_2, S_1) = HV(S_1 + S_2) - HV(S_1) \quad (5.5)$$



**Figure 5.11 :** 2D illustration of two fictitious Pareto front representations to be compared.

**Hyper-volume Ratio,  $HR(S_1, S_2)$  (Van Veldhuizen, 1999):** is another binary implementation for hypervolume metric that gives the proportion of domination area hypervolumes of the two Pareto fronts, (5.6),

$$HR(S_1, S_2) = \frac{HV(S_1)}{HV(S_2)} \quad (5.6)$$

**Spacing (Spc) (Coello Coello, 2007; , Schott, 1995):** is a measure of homogeneity of the Pareto front. Spacing numerically describes the spread of the vectors in the non-dominated set.

$$Spc \triangleq \sqrt{\frac{1}{|S|-1} \sum_{i=1}^{|S|} (\bar{d} - d_i)^2} \quad (5.7)$$

where  $d_i = \min_j (|C_2^i(x) - C_2^j(x)| + |C_3^i(x) - C_3^j(x)|)$ ,  $i, j = 1, \dots, |S|$ ,  $\bar{d}$  is the mean of all  $d_i$  and  $C_k^i$  stands for the objective value corresponding to the  $k^{th}$  function for the  $i^{th}$  individual in the archive.

**Overall Non-dominated Vector Generation (ONVG) (Van Veldhuizen, 1999; Van Veldhuizen and Lamont, 1999):** measures the total number of non-dominated vectors found during algorithm execution, (5.8),

$$\text{ONVG} = |S| \quad (5.8)$$

**$M_3^*$  Spread ( $M_3^*$ ) (Zitzler, 1999):** is a unary spread metric that is used to evaluate the ability of the algorithm in extreme conditions. The algorithm sums up the squares of the largest distances in different objectives (Fig. 9),

$$M_3^*(S) = \sqrt{(\sum_{i=1}^{\delta} \max\{\|\mathbf{u}_i - \mathbf{v}_i\| \mid s, t \in S\})} \quad (5.9)$$

where  $\|\cdot\|$  stands for a way of measuring distance. In this study,  $M_3^*$  Spread distances are evaluated in Euclidean norm.



## **6. ENHANCED ORDER BASED SINGLE LEAP-BIG BANG BIG CRUNCH OPTIMIZATION APPROACH TO MULTI-OBJECTIVE AIRPORT GATE ASSIGNMENT PROBLEM**

### **6.1 Introduction**

Gate assignment problem (GAP) is well studied in the literature and consequently, there are many proposed problem formulations and solution techniques. Though the basic constraints and objectives are easily perceived, the problem has many interactions with other resources such as the number of gates, airport topology, flight schedules, distances to baggage claim areas, etc. Therefore, GAPs are even more complicated than most other traditional scheduling problems (Dorndorf et al, 2007). Moreover, as the air traffic becomes more demanding, the grandeur of the solution space gets even larger; in return, this makes traditional binary integer techniques practically inapplicable. In those cases, nature inspired computing techniques became a good alternative for GAPs.

A practical formulation for the GAP could have multiple objectives and a corresponding solution technique should handle possible large solution spaces. For instance, a central European airport hosts up to a thousand flights over approximately a hundred gates summing up to nearly  $1000^{100}$  possible solution candidates. There are many multi-objective formulations reported in literature (Teodorovic & Guberinic, 1984; Teodorovic & Stojkovic, 1990; Ding et al, 2004; Ding et al, 2004; Ding et al, 2005; Hu & Paulo, 2007; Chang, 1994; Dorndorf, 2002; Yan & Huo, 2001; Zhu et al, 2003; Wei & Liu, 2007); however, most of the proposed formulations either fuse the preferences into a single objective function and omit compromise solutions or use classical integer programming techniques in relatively small problem instances. Drexl and Nikulin (2008) propose a multi-objective problem formulation with three objectives and use pareto simulated annealing method to obtain a pareto front of solutions. This is the first study using a multi-objective evolutionary approach capable of handling a very large scale problem. The objectives are to minimize the number of ungated flights and the total passenger

walking distances as well as to maximize the total gate assignment preferences. In that study, a lexicographic approach has been used to minimize first the ungated flights with an optimal greedy method (Xu and Bailey, 2001). Next, the problem solving has been carried out over the remaining two objectives, starting from the assignment list obtained after the application of the greedy method.

In this chapter, problem of maximizing gate duration in chapter 4 is expanded to include passenger walking distances and gate preferences. The solution technique using plane ordering method has been enhanced to generate compromise solutions of the pareto front. Besides, a real problem instance generation method is developed and used for the experiments. The algorithm is also verified on data collected from the Atatürk Airport of İstanbul.

The chapter is organized as follows: The GAP mathematical model is given in the next section. The details of the new method are presented in section 3 which also includes a brief overview of the previously developed multi-objective gate assignment methods reported in the literature. The simulation results based on artificial and real data set are then given in section 4. Finally, the concluding remarks and discussions are done in section 5.

## 6.2 Problem Formulation

The problem of airport gate assignment is formulated as a multi-objective optimization problem. The objectives introduced include the maximization of gating duration, minimizing the total walking distance and maximizing the gate preferences.

The parameters are defined as follows:

$F$  : set of flights arriving at and/or departing from the airport

$G$  : set of available gates at the airport

$N_f$  : total number of flights, i.e.  $N_f = |F|$

$N_g$  : total number of gates, i.e.  $N_g = |G|$

$N_t$ : number of time slots in a day (depends on time slot length  $n$ )

$T_{A(i)}$  : arrival time of flight  $i$

$T_{D(i)}$  : departure time of flight  $i$

$w_{k,l}$  : walking distance for passengers from gate  $k$  to gate  $l$

$f_{i,j}$  : number of passengers transferring from flight  $i$  to flight  $j$

$pf_i$  : preference value for flight  $i$

$pg_{i,k}$  : normalized preference value of assigning flight  $i$  to gate  $k$

$M_c$  : ( $N_g \times N_t$ ) matrix of assignments (gate assignments)

where,  $M_c(i, j) = U$ , ( $U=1, \dots, N_f$ ), if the gate  $i$  is assigned at time slot  $j$  to the  $U_{th}$  flight,  $M_c(i, j) = 0$ , if otherwise.

In addition, two dummy gates are introduced as in Drexl and Nikulin (2008). Gate  $N_g+1$  represents the apron and the gate  $N_g+2$  represents the entrance / exit of the airport. The variable  $x_{i,k} = 1$  denotes that flight  $i$  is assigned to gate  $k$ , such that  $1 \leq k \leq N_g + 1$ , and  $x_{i,k} = 0$  otherwise. Then, the objective functions can be defined (all in minimizing form) as follows:

**O<sub>1</sub>**: Gate duration maximization (negative function is minimized)

$$\begin{aligned} \min C_1 &= -\sum_{k=1}^{N_g} \sum_{l=1}^{N_t} any(M_c(k, l)), & (6.1) \\ &any(M_c(k, l) = 1 \text{ if } M_c(k, l) \neq 0) \\ &any(M_c(k, l) = 0 \text{ if otherwise.} \end{aligned}$$

**O<sub>2</sub>**: Walking distance minimization

$$\begin{aligned} \min C_2 &= \sum_{i=1}^{N_f} \sum_{j=1}^{N_f} \sum_{k=1}^{N_g+1} \sum_{l=1}^{N_g+1} f_{i,j} w_{k,l} x_{i,k} x_{j,l} + \sum_{i=1}^{N_f} \sum_{k=1}^{N_g+1} f_{N_g+2,i} w_{N_g+2,k} x_{i,k} + \\ &\sum_{i=1}^{N_f} \sum_{k=1}^{N_g+1} f_{i,N_g+2} w_{k,N_g+2} x_{i,k} & (6.2) \end{aligned}$$

**O<sub>3</sub>**: Preference maximization (negative function is minimized)

$$\min C_3 = -\sum_{i=1}^{N_f} \sum_{k=1}^{N_g+1} pf_i pg_{i,k} x_{i,k} \quad (6.3)$$

Objective function given in (6.1) is to maximize gate duration, which is total occupation time of the gates allocated for all flights within a day. This objective function is already studied in Chapter 4. Gate duration maximization is first studied in Genç et al. (2011, 2012) and reported to be the most important criterion since it is

highly correlated with the revenue obtained. The second objective represents the total passenger walking distance minimization and it is another most commonly used objective function. The third objective represents the total flight to gate assignment preference. Preference is a general term mathematically covering airline dependent priorities.

An operation day is quantized with  $n$  minute long time intervals in order to measure the occupation density of the gates (see Chapter 4).

The formal definitions for the constraints introduced in Chapter 4 are given next:

- 1) One gate can only accommodate a single aircraft at a time; and therefore, two flights must not be assigned to the same gate if their staying times overlap in time (Dorndorf et al, 2007). This can be expressed as **(6.4)**

$$x_{i,k} x_{j,k} (T_{D(j)} - T_{A(i)})(T_{D(i)} - T_{A(j)}) \leq 0 \quad 1 \leq i, j \leq N_f, k \neq N_g + 1 \quad (6.4)$$

- 2) Every flight must be assigned to only one gate (or apron) **(6.5)**

$$\sum_{k=1}^{N_g+1} x_{i,k} = 1, \quad 1 \leq i \leq N_f \quad (6.5)$$

The constraint given in (2.5) is not valid for assignment problems that allow the movement of planes between gates during their stay at the airport. This is not permitted in the formulation of this problem.

### 6.3 Multi-Objective Gate Assignment Problem Solution Techniques

Drexel and Nikulin (2008) used the well-studied three objectives; namely, minimizing the number of planes assigned to apron area, minimizing total walking distances and maximizing preferences formulated in their GAP. Passenger walking distance minimization and preference maximization are given in **(6.2)** and **(6.3)**, respectively. The objective of minimizing the number of planes allocated to apron is formulated as in **(6.6)**,

$$\min C_1 = \sum_{k=1}^{N_g+1} x_{i,N_g+1}, \quad (6.6)$$

An implementation of the Pareto Simulated Approach (PSA), which is the multi-objective adaptation of the simulated annealing algorithm, is used to solve their



MOGAP in Drexl and Nikulin (2008). In that study, the designer has to define many parameters such as the search space, neighbor generation method, neighbor acceptance criterion, cooling function and the stopping temperature. The main distinction with the single objective case is the design of the acceptance rule. The idea in PSA acceptance probability is a local aggregation of all objectives with the weighted Tchebycheff function, (6.7) or weighted linear function with reference to the current solution (6.8)

$$P_1(x, x', \lambda, T_i) = \min \left\{ 1, \exp \left( - \max_{k=1, \dots, \delta} \frac{\lambda_k [C_k(x') - C_k(x)]}{T_i} \right) \right\} \quad (6.7)$$

$$P_2(x, x', \lambda, T_i) = \min \left\{ 1, \exp \left( - \sum_{k=1}^{\delta} \frac{\lambda_k [C_k(x') - C_k(x)]}{T_i} \right) \right\} \quad (6.8)$$

where  $\delta$  is the number of objective functions and  $T_i$  is the temperature value associated with  $i^{th}$  iteration.  $\lambda$  is the weighting coefficient that changes at each iteration in order to increase the probability of moving current solution away from its closest neighbor.

In Drexl and Nikulin (2008) the objective of minimizing the number of planes allocated to apron lexicographically dominates the other objectives. Hence, that objective is firstly tried to be minimized then PSA is applied to the remaining two objectives. The initial gate assignment solution, which is the solution for minimum planes assigned to the apron as given in (6.6), is obtained using the optimal greedy allocation method (Ding et al, 2005). Then,  $N_a$  distinct agents are generated by applying apron moves (Ding et al, 2005) to the greedy solution  $N_a$  times to the initial population. In the following iterations, the algorithm searches for neighbors by using simple moves on the assignment list. This neighboring search approach consists of three moves (Ding et al, 2005):

**The Insert Move:** Move a single flight to a gate other than the one it has been currently assigned.

**The Interval Exchange Move:** Exchange two flight intervals in the current assignment. A flight interval consists of one or more consecutive flights in one gate.

**The Apron Exchange Move:** Exchange one flight which has been assigned to the apron with a flight that has been currently assigned to a gate.

The types of neighboring moves can be further studied from the single objective GAP application in (Ding et al, 2005; Xu and Bailey, 2001). The acceptance criterion is given in (6.9)

$$P(x, x', \lambda, Ti) = \min \times \{1, \exp (-a \times \max_{k=2,3} \frac{\lambda_k[\tilde{f}_k(x') - \tilde{f}_k(x)]}{b \times Ti})\} \quad (6.9)$$

where  $\tilde{f}_k$  is the normalized function value and  $a = b = 1$ . The initial temperature is selected as in (6.10)

$$T_0 = 2 * N_f \quad (6.10)$$

and the cooling schedule is expressed as in (6.11),

$$T_i = 0.998^i * T_0 \quad (6.11)$$

where  $i$  stands for the iteration number. Algorithm continues until the temperature decreases below  $10^{-4}$ .

To the best of our knowledge, this work is unique in comprising the following properties:

- I. There are multiple objectives.
- II. The algorithm is solved with multi-objective techniques to obtain a pareto front in objective space.
- III. Multi-objective optimization algorithm is a (evolutionary) nature inspired computation technique.

However, there are some shortcomings in the solution approach given in Drexl and Nikulin (2008) such as

- Acceptance criterion is formulated assuming all the objectives have to be minimized by taking the inverse or the negative of the third objective function (that is originally a maximizing one).
- The exponential term approaches to zero since the starting temperature is in the orders of hundreds to thousands and constant  $b$  is equal to one. As final iterations are reached the candidate acquires a high probability of selection.
- In under-constrained situations, where there is no need to assign any plane to the apron, the method is unable to generate a solution.

- The algorithm focuses on modifying the final assignment list by neighboring search moves. In practical situations, the running procedure can be cumbersome due to the constraints, which in turn, leads to very long computational run times. In most cases, this is neither favorable nor acceptable since frequently occurring flight delays pin down a quick reconfiguration of the gate assignment list.
- Though the problem is formulated in three-dimensional objective space, highly dominant objective of minimizing the number of planes allocated to apron is optimized first, then the problem is solved for a two-dimensional pareto front. There is no concern or information on the objective score corresponding to the number of planes assigned to the apron for the pareto optimal members.

### **6.3.1 Enhanced order based SL-BBBC algorithm (eSL-BBBC):**

In their recent work, Genç et al. (2011, 2012) defined an objective function to maximize gate time duration which is the total time of the gates allocated for all flights of a day. This objective function is given in (6.1). The proposed solution strategy, SL-BBBC algorithm is reviewed in Chapter 4.

One of the main contributions of this paper is the Enhanced Order Based Single Leap-Big Bang Big Crunch algorithm (eSL- BBCC) algorithm for the solution of the MOGAP proposed in (6.1)-(6.3). The algorithm has its origins from the aforementioned single objective counterpart. It starts with generating handling order of planes by GTMA heuristic. However, in the assignment phase, objective preferences are considered; that is, the assignments are done according to minimization of walking distances or maximization of preferences. Gate duration maximization is inherently included by these two functions since all the gates equally contribute to the objective score. The non-dominated members obtained through the algorithm are stored in the archive population. The multi-objective version of the above algorithm can be briefly summarized as follows and it is illustrated in **Figure 6.6**:

STEP 1: Apply Ground Time Maximization Algorithm (GTMA) and find an assignment list.

STEP 2: Store the objective function values in a vector form.

STEP 3: Log the order in which the planes are assigned.

STEP 4: Apply Enhanced Order Based Single Leap-Big Bang Big Crunch Algorithm (eSL-BBBC) to change handling order, at the same time, find the pareto optimal solution candidates by using dominance evaluation.

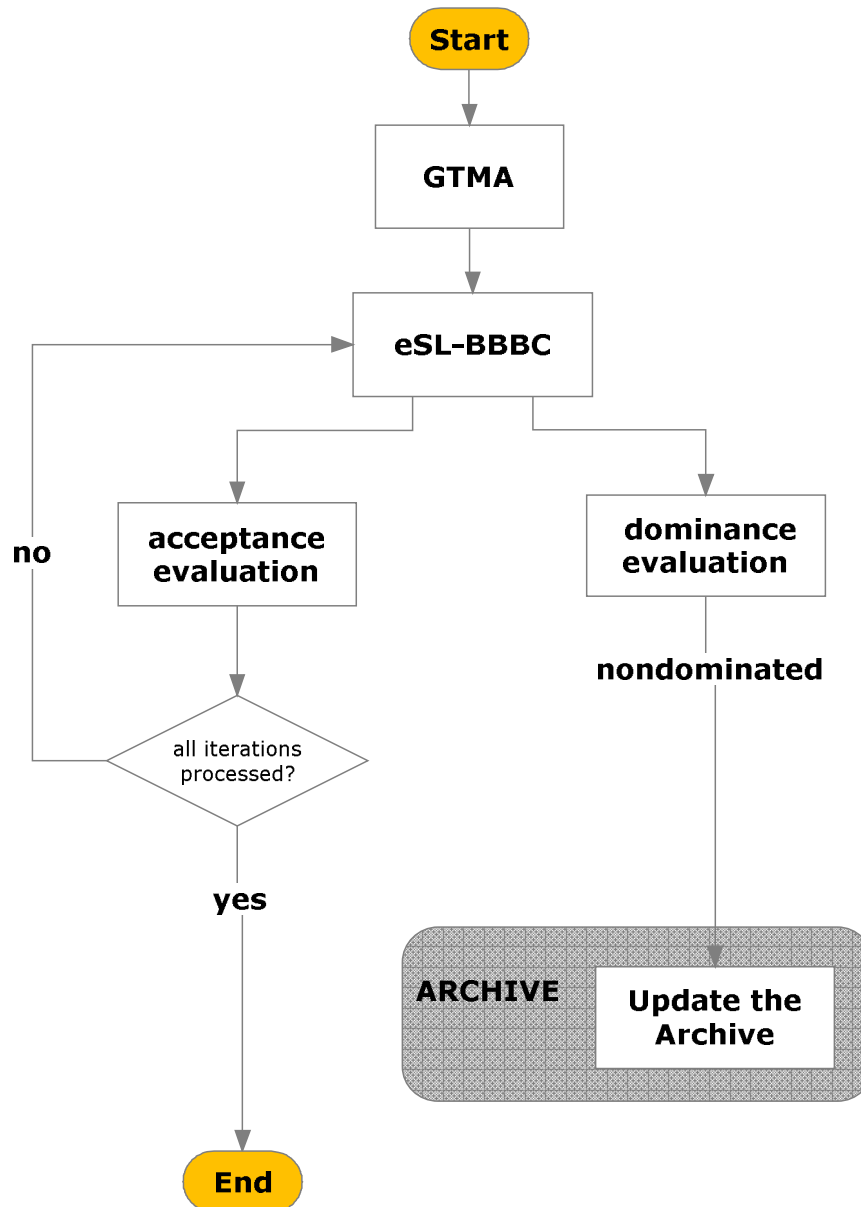


Figure 6.1 : Main loop of the MOGAP algorithm.

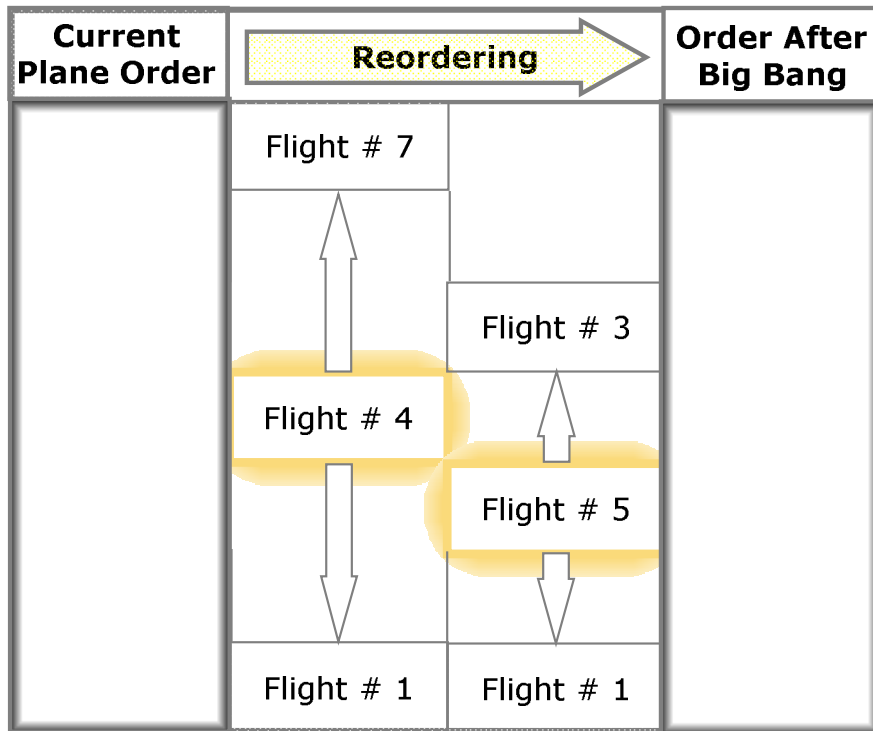
### 6.3.1.1 Creating the initial population

GTMA creates a single order for the plane handling process. Then, the initial population of solutions is generated from this order by swapping the plane positions

randomly. Hence, unlike the case in Genç et al. (2012), the algorithm can progress by relying on more than one solution candidate. However, the solution candidates do not interact with each other in the primary population.

### 6.3.1.2 Neighbor generation

The solution candidates (the individuals of the population) are considered to be plane orders. To generate neighbors, a modification is done on the plane orders by the utilization of SL-BBBC. The big bang phase of the SL-BBBC algorithm is implemented as in Genç et al. (2012). The effect of big bang resembles the  $N$ -swap mutation operator where the number of swappings, and the distance of the swapping members are controlled by the explosion strength of “big bang” phase (Figure 6.2).



**Figure 6.2 :** Interchanging the order of  $N = 2$  random flight pairs with random distances away from random centers in the list. The first bang centers the 4<sup>th</sup> flight and explosion strength is 3 units, whereas the second flight centers the 5<sup>th</sup> flight and explosion strength is 2 units.

### 6.3.1.3 Assignment of planes

The plane ordering process constitutes a basis for the allocation of the flights with long staying times in the first place. In the proposed algorithm, planes are assigned with respect to the order logged in the solution candidate as well as their specific

objective function preferences. For instance, each plane is assigned to the nearest available gate for transferring and embarking / disembarking passengers for passenger walking minimization and the plane is assigned to the gate having the most preference value for the flight for preference maximization. Besides, each plane is assigned according to one objective function that is selected randomly between the two. Then, in most general sense, the algorithm creates  $(\delta + \beta)$  offspring for any solution candidate, where  $\delta$  is the number of objective functions and  $\beta$  is the number of objective function combinations. In this study  $\delta$  and  $\beta$  are chosen to be equal to 2 and 1, respectively. Details of the algorithm tailored for the MOGAP are given in **Figure 6.3**.

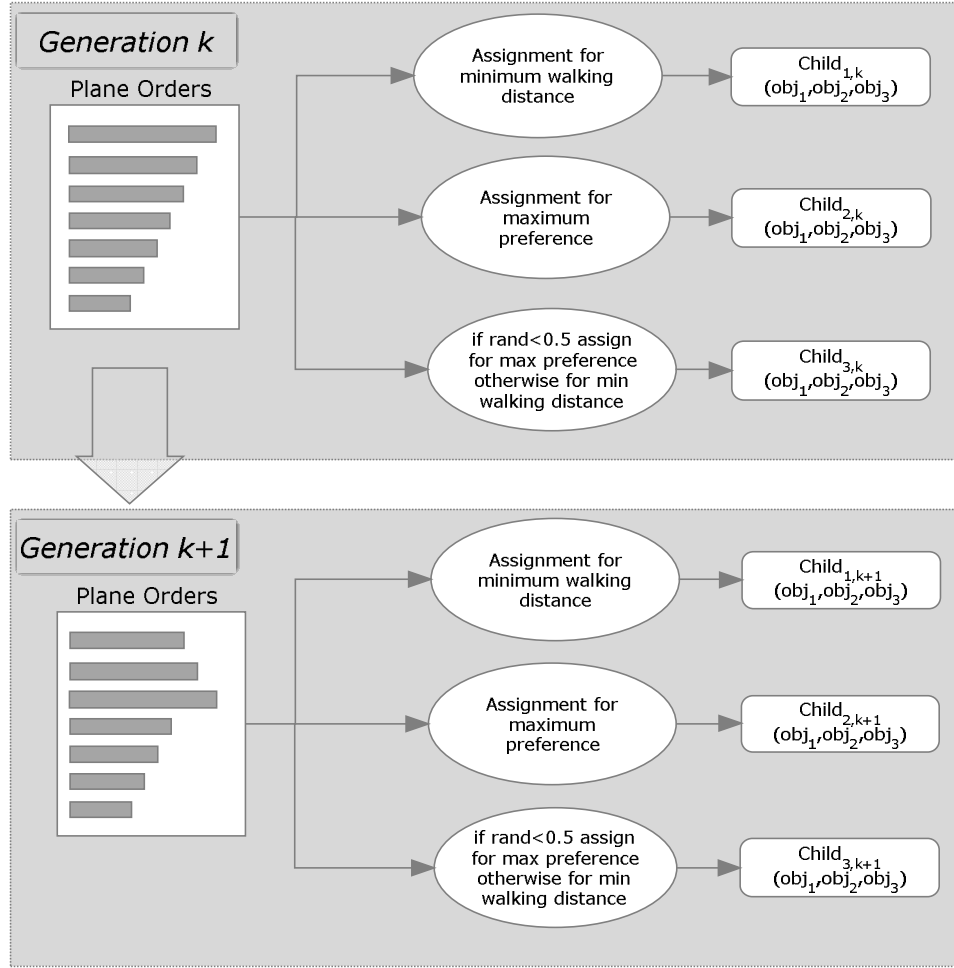
#### 6.3.1.4 Acceptance of the neighbors

The  $(\delta + \beta)$  offspring dominance status defines the acceptance probability of their parent order. Each new created child is compared with the previously generated child using the same objective function preference. They can be in one of the three states cited below with respect to other in terms of dominance relation (**Figure 6.3**):

1. child generated at iteration  $k$  dominates the one created at iteration  $k+1$ :  
 $Child_k \preceq Child_{k+1}$ ,  
then gets the score of  $\mu_{j1}$
2. child generated at iteration  $k$  is dominated by the one created at iteration  $k+1$ :  
 $Child_{k+1} \preceq Child_k$ ,  
then gets the score of  $\mu_{j2}$
3. they are both non-dominated with respect to each other, that gets the score of  $\mu_{j3}$

Here,  $\mu_{ji}$  is the normalized acceptance score assigned in conjunction with the child's dominance relation. Note that  $j \in Z$ ,  $1 \leq j \leq \delta + \beta$  and  $\mu_{j2} \geq \mu_{j3} \geq \mu_{j1}$ . Then the acceptance probability for an assignment order representation becomes, **(6.12)**,

$$P(x') = \sum_{i=1}^3 \sum_{j=1}^{\delta + \beta} \gamma_i \mu_{ji} \quad (6.12)$$



**Figure 6.3 :** Algorithm progress: Each solution representation is decoded into three objective space vectors.  $Child_{1,k}$  is created favoring minimum walking distance;  $Child_{2,k}$  is created favoring flight to gate preferences.  $Child_{3,k}$  is created by randomly favoring the two objective functions. Acceptance of the plane order in generation  $k+1$  depends on the dominance levels of ( $Child_{1,k+1}$  to  $Child_{1,k}$ ), ( $Child_{2,k+1}$  to  $Child_{2,k}$ ) and ( $Child_{3,k+1}$  to  $Child_{3,k}$ ).

where  $y_i$  is the binary decision variable set to 1 if  $i^{th}$  dominance relation holds; otherwise, it is reset to 0.

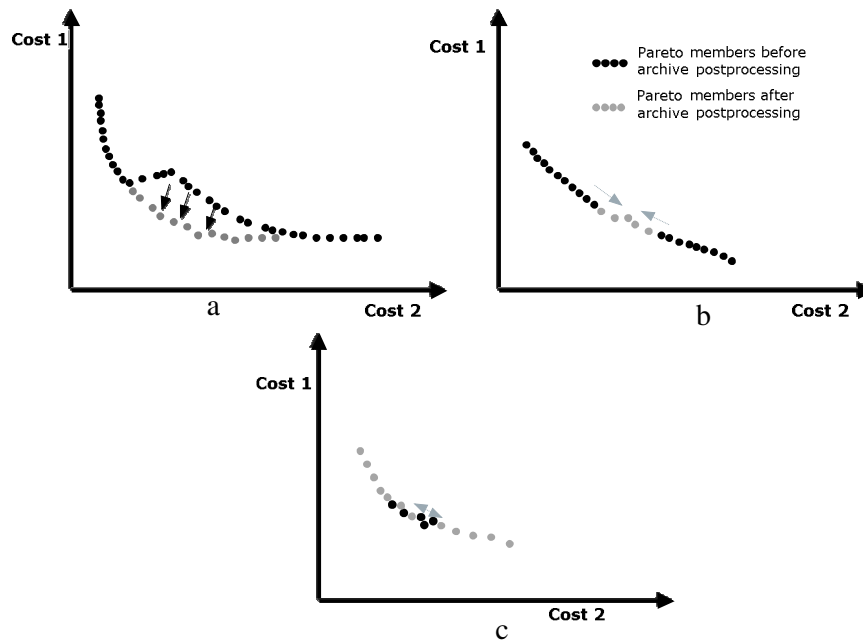
Since the dominance relations are mutually exclusive, the following relation should hold

$$\sum_{i=1}^3 y_i = 1 \quad (6.13)$$

In this formulation, there is no explicit term that gradually decreases the probability of acceptance as in (6.9). However, the probability of finding a dominating (or non-dominated) individual decreases as the iterations elapse; therefore, the acceptance probability inherently decreases.

### 6.3.1.5 Archive postprocessing

All the visited members are candidate for archive collection. The algorithm run ends up with a bunch of members that are nondominated and these members represent the pareto front. These pareto front members may be not homogenously scattered and / or not adequately informative. Finally, during archive postprocessing, the possible gaps among pareto front members are tried to be filled by homogenously scattered new members and pareto front itself is tried to be shifted into a better position hence optimizing the total related cost values (**Figure 6.4**).



**Figure 6.4** : The effect of archive postprocessing: (a) Pareto front shift, (b) Repairing gaps, (c) Homogenizing the distribution.

Archive postprocessing algorithm is given in **Table 6.1**.

**Table 6.1:** Archive Postprocessing Algorithm.

```

for i = 1 :  $S_{arc}$  do
    Copy archive to archive mating list
    while archive mating list is not empty
        Select two random individuals to mate
        Discard these individuals from the archive mating list
        Apply PMX
        Store the offspring in offspring population
    end
    Add offspring population to the archive
    Eliminate the dominated members in archive
end

```



### 6.3.1.6 Stopping criterion

The stopping criterion of the algorithm can be chosen as the maximum number of iterations or function evaluations, a convergence measure for the population or any combination of the above-mentioned criteria. In this study, function evaluation number is used as the stopping criterion of the algorithm for fair comparison in experiments.

The complete algorithm steps can be given as in **Table 6.2**.

**Table 6.2** : MOGAP solution.

|   |
|---|
| <p><i>Apply GTMA given in Algorithm 1</i></p> <p><i>Calculate the initial cost vector and then apply the following loops</i></p> <p><b>for</b> <math>i = 1</math> : number of iterations (<math>N_i</math>)</p> <p>    <b>for</b> <math>k = 1</math> : number of individuals in the population (<math>N_p</math>)</p> <p>        i. <i>Apply a big bang step to get a new order of planes</i></p> <p>        ii. <i>Assign the planes for minimum walking distance and obtain child<math>_{1,k}</math></i></p> <p>        iii. <i>Assign the planes for maximum preference and obtain child<math>_{2,k}</math></i></p> <p>        iv. <i>Assign the planes for maximum preference or minimum walking distance that is selected randomly and obtain child<math>_{3,k}</math></i></p> <p>        v. <i>Calculate cost vector for all children and check whether they will be added to archive or not.</i></p> <p>        vi. <i>Calculate the acceptance probability <math>P(x')</math> of the new solution using Eq. (3.7).</i></p> <p>        vii. <i>Accept the new plane order with probability equals to <math>P(x')</math></i></p> <p>    <b>end</b></p> <p><b>end</b></p> <p><i>Perform archive postprocessing given in Algorithm 2.</i></p> |
|---|

Acceptance criterion for the neighbors is designed to be quite simple and self-adjusting the selective pressure parameter. Initial population is generated independent of the planes assigned to apron or gate. The neighbors are generated using plane assignment orders, not the final assignment list. Finally, although it is also highly related to the remaining two objective functions, the first objective function is optimized within the problem. Therefore, the problems existing in PSA based MOGAP solution reported in Drexl and Nikulin (2008) are eliminated or cured in the approach to the problem and in the algorithm presented in this chapter.

## 6.4 Simulation Results

In order to make realistic and credible simulations on the newly developed algorithm, the problem instance must be designed veraciously. For this purpose, a generator engine is designed to provide a quasi-realistic airport plane and pedestrian traffic data, which is the second main contribution of this Chapter. In the succeeding subchapter, this engine including airport flight generation (detailed in chapter 4), airport walking distance, passenger flow and preference assignments modules will be discussed in detail. Then, the MOGAP formulated in Eq. (2.1-2.3) will be solved for both the artificial data formed by the above problem instance generator, and real flight data obtained from İstanbul Atatürk Airport.

### 6.4.1 Problem instance generation

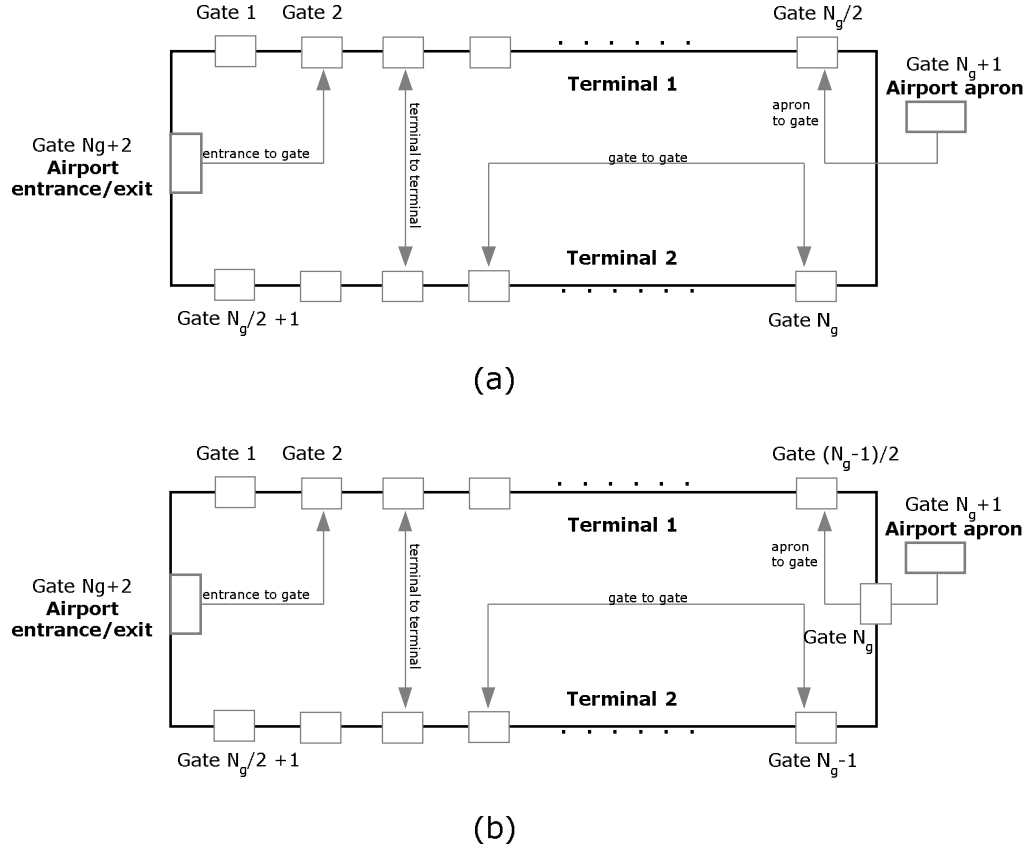
Problem instance consists of a complete flight schedule with arrival and departure times of every single flight, airport walking distances, number of passenger transfers between the flights and the preferences of flight to gate assignments.

#### 6.4.1.1 Flight generation

Flight generation is the same as given in Chapter 4.

#### 6.4.1.2 Airport topology and walking distances

Airport topology and walking distance model is the generalized version of airport gate and walking distances model given in Drexler and Nikulin (2008). The layout of a representative airport consists of two parallel terminals and symmetrically located  $N_g$  gates. Since there are two parallel terminals, each terminal has  $(N_g / 2)$  gates (**Figure 6.5a**). If the number of gates is odd, remaining gate is located in between gates  $(N_g - 1) / 2$  and  $(N_g - 1)$  and between the two terminals (**Figure 6.5b**). There are two more dummy gates: one to represent all the parking places of the apron, gate  $N_g + 1$ , and one to represent airport entrance / exit (assumed to be the same place), gate  $N_g + 2$ .



**Figure 6.5** : The layout of a representative airport.

Distance between gates is measured according to the Manhattan metric, and then passengers are allowed to move only horizontally and vertically. The distance matrix  $W$  for  $N_g$  gates is defined as (6.14),

$$w_{i,j} = \begin{cases} 0 & \text{if } i = j \\ g_{dist} * |i - j| & \text{if condition I applies} \\ g_{dist} * \left| \max(i, j) - \frac{m}{2} - \min(i, j) \right| + t_{dist} & \text{if condition II applies} \end{cases} \quad (6.14)$$

$$\text{I: } \left( i < \frac{N_g}{2} + 1 \right) \wedge \left( j \leq \frac{N_g}{2} \right) \vee \left( i > \frac{N_g}{2} \right) \wedge \left( j \geq \frac{N_g}{2} + 1 \right)$$

$$\text{II: } \left( i < \frac{N_g}{2} + 1 \right) \wedge \left( j \geq \frac{N_g}{2} + 1 \right) \vee \left( i > \frac{N_g}{2} \right) \wedge \left( j \leq \frac{N_g}{2} \right)$$

where  $t_{dist}$  is the distance between two terminals,  $g_{dist}$  is the distance between two neighboring gates belonging to the same terminal. Additionally, the walking

distances between the gates and the airport entrance / exit are assumed to be symmetric as given in (6.15)

$$w_{N_g+2,i} = w_{i,N_g+2} = i + P_{e/e} \quad i = 1, \dots, \dots, \frac{N_g}{2} \quad (6.15)$$

where  $P_{e/e}$  is the extra walking distance added for entrance / exit gate. The walking distances between the apron and the airport gates are as follows:

$$w_{N_g+1,i} = w_{i,N_g+1} = i + P_a \quad i = 1, \dots, \dots, \frac{N_g}{2} \quad (6.16)$$

That is, an assignment to the apron is penalized with an extra penalty,  $P_a$ . Moreover, if passengers are walking from a plane assigned to the apron to another one that is also in the apron, then a walking distance,  $P_{a,a}$ , is added which is missing in Drexl and Nikulin (2008).

#### 6.4.1.3 Passenger flow model

Passenger flow model given in Drexl and Nikulin (2008) has many deficiencies that can be summarized as follows:

- A plane can host up to half a thousand passengers in the model and the number of passengers rises as the number of planes or time horizon increases. In general, airplanes carry between 100 to 300 passengers and a maximum of about 800 (Url-3).
- The number of boarding passengers is not equal to the number of passengers deplaning.
- All the planes are assumed to have the same passenger capacity; there are not any fluctuations in passenger numbers due to physical conditions.
- All the planes have similar gating durations. No turnaround time impact is allowed.
- Deplaning passengers can be transferred to any flight departing after a while from their arrival to the airport. However, in practice, the passengers may transfer to only a small portion of the available flights.
- In real life, transfer passengers are mostly booked to the soonest departing flight and the probability of transfer decreases as time elapses. Even if there is a connected flight long after arriving at the airport, this should not increase walking distance penalty. This is not the case in the reported model.

In this chapter, a high fidelity passenger flow model is proposed. Details of the model parameters and remedy options for the previously reported literature are given in subsequent sections.

**Number of Passengers:** Scheduled plane gate duration depends on the turnaround time of the planes. A major component of turnaround time is the passenger boarding time (Horstmeier and Haan, (2001) provide detailed analyses of all the components in an aircraft turn-around time) and boarding time is directly related to the number of passengers. Then, in a realistic model, as the number of passengers increase, gating durations get longer. The flow model proposed in this work has passenger counts distributed uniformly in the interval  $[c - \gamma, c + \gamma]$ , where  $c$  is the mean passenger count for the scheduled flight and  $\gamma$  is a normalized constant introducing randomness. Mean passenger count for flight  $i$  is obtained by multiplying scheduled staying time with the average passenger count for an hour of turnaround, (6.17),

$$c_i = (T_{D(i)} - T_{A(i)}) / (60/n) * \tau \quad (6.17)$$

where  $c_i$  is the mean passenger count of flight  $i$ , and  $\tau$  is the average passenger count for a plane.  $\tau$  is typically in between 150 and 250 (Bazargan, 2004; Url-3)

**Consistency of the passenger flow model:** After the arrival of an airplane at an airport, all the passengers are either transferred to another plane or disembarked through the exit gate. Similarly, before departure, all the passengers are either transferred from another flight or embarked. Hence, the number of boarding passengers must be equal to the number of passengers deplaning.

**Transferring passenger behaviors:** Airlines utilize hub and spoke systems to provide connections between city pairs in their network. Then, the probability of connections increases; however, the passengers are not transferred to all the flights homogenously but to only a small subset (percent of transferred flights,  $F_{tp}$ ) of them.

The number of passengers disembarking after deplaning is generally more than transferring passengers to a specific flight. Percent of transferring passengers ( $P_{tp}$ ) is defined typically in between 30% - 80%. These passengers are transferred to flights and remaining passengers are assumed to be disembarked.

Departure of the plane  $j$  must be at least  $B_{min}$  time slots later than arrival of the plane  $i$  in order to be able to transfer passengers from flight  $i$  to flight  $j$ , (6.18)

$$T_D(j) \geq T_A(i) + B_{min} \quad (6.18)$$

As the time interval between flights increases, the probability of transfers decreases proportionally with  $(T_D(j) - T_A(i))$ . Moreover, if the following condition holds

$$T_D(j) \geq T_A(i) + B_{max} \quad (6.19)$$

where  $B_{max} \gg B_{min}$ , then the walking distance penalty will no longer be important as the time interval between two flights is too long that passengers will probably walk not only between the gates but also inside or outside the airport.

#### 6.4.1.4 Preference model

Flight to gate assignment preferences depends on many physical, economical and even political constraints. Instead of separately modeling all, the preference of a specific flight and assigning a flight to a specific gate is considered.

Flights with more passengers and high security flights have higher preference values. Since both of these categories need more turnaround times (Horstmeier and Haan, 2001), preference value of each flight can be modeled directly related to the gate duration (Note that, in Drexl and Nikulin (2008), flight preferences are generated randomly). The vector of preferences  $\mathbf{V} = \langle v_i \rangle$  is generated from normal distribution having mean  $\mu_p$  and standard deviation  $\sigma_p$ .  $\mu_p$  is calculated by normalizing each scheduled gate duration with the maximum gate duration of the day. The preference value  $u_{i,k}$  of assigning flight  $i$  to gate  $k$  is randomly generated within the interval (0,1). Then, the overall preference of assigning flight  $i$  to gate  $k$  becomes  $v_i * u_{ik}$ .

#### 6.4.2 Experiments on artificial data

Multi-objective gate assignment algorithms are compared by using artificial data of the test data generator given in the preceding chapter. In the experiments, Moderate dataset and high gate demand distributed uniformly data sets that are introduced in Chapter 4 have been used with slight modifications and additions. The parameters used to generate these sets of data are given in **Table 6.3**. In generating a dataset, a reasonable set of values have been chosen; there is no effort in finding the algorithm performances in the whole range of parameters. This task is left as a future work.

**Table 6.3** : Parameters of the artificial data sets.

| Parameter     | Moderate Demand | High Demand | Parameter  | Moderate Demand | High Demand |
|---------------|-----------------|-------------|------------|-----------------|-------------|
| $N_g$         | 15              | 20          | $g_{dist}$ | 1               | 1           |
| $n$           | 1               | 1           | $P_{e/e}$  | 2               | 2           |
| $N_t$         | 265             | 265         | $P_a$      | 5               | 2           |
| $d$           | 0.7             | 0.94        | $P_{a,a}$  | 5               | 5           |
| $p_1$         | 0.1             | 0.0         | $\tau$     | 150             | 200         |
| $p_2$         | 0.1             | 0.0         | $F_{tp}$   | 0.25            | 0.2         |
| $\mu_{fg}$    | 30              | 50          | $P_{tp}$   | 0.6             | 0.75        |
| $\sigma_{fg}$ | 20              | 40          | $B_{min}$  | 20              | 30          |
| $t_{dist}$    | 2               | 2           | $B_{max}$  | 100             | 120         |

Every experiment is carried independently for 30 times to produce reliable statistics since the running algorithms have stochastic nature. 30 runs are assumed to be enough to obtain credible results in the literature (Huang et al, 2007; Zhang et al, 2009; Deb et al, 2002). For a fair comparison, remaining test parameters are selected similarly as given in Drexl and Nikulin (2008):

- Both algorithms are allowed to perform same number of function evaluations. Number of function evaluations ( $N_{fe}$ ) is selected to be 7248 where the temperature decreases below  $10^{-4}$ .
- Time horizon for the flight data is 4 hours and 30 minutes and the length of one time slot,  $n$ , is equal to be 1 minute.
- Only the second and third objective functions are considered. In both works, the first objective function is strongly correlated with the other two. Hence it is rational to compare pareto fronts in 2-dimension. Nevertheless, the results of the first objective functions will also be reported.
- PSA based MOGAP have identical parameter settings as suggested in the original study.

No single metric can entirely capture total algorithm performance (Coello Coello et al, 2007). Performance metrics can be classified in two groups as convergence metrics and distribution (and spread) metrics (Yu and Gen, 2010). In this study, four convergence metrics and three distribution metrics are used. Algorithm performances with respect to the selected metrics are reported in **Table 6.4** and **Table 6.5** for the moderate and high demand data sets, respectively. In the tables, seven different

datasets representing the days of the week are used and daily average values for the performance metrics are given. The last column of the specific algorithm reports the average of the weekly scores.

The performance metrics are summarized in Chapter 5.8.

**Table 6.4 :** Performance metrics for moderate demand data set (For binary metrics  $CS$ ,  $D$  and  $HR$ ,  $S_I$  is the pareto optimal set of the related algorithm).

| Performance Measure | PSA based MOGAP |       |       |       |       |       |       |       | eSL-BBBC based MOGAP |       |       |       |       |       |       |       |
|---------------------|-----------------|-------|-------|-------|-------|-------|-------|-------|----------------------|-------|-------|-------|-------|-------|-------|-------|
|                     | Day 1           | Day 2 | Day 3 | Day 4 | Day 5 | Day 6 | Day 7 | Av    | Day 1                | Day 2 | Day 3 | Day 4 | Day 5 | Day 6 | Day 7 | Av    |
| $CS(S_I, S_2)$      | 2,90            | 4,57  | 2,10  | 3,97  | 5,30  | 0,87  | 2,47  | 3,17  | 12,60                | 8,83  | 11,83 | 11,33 | 10,27 | 9,13  | 13,37 | 11,05 |
| $D(S_I, S_2)$       | 0,00            | 0,00  | 0,00  | 0,00  | 0,00  | 0,00  | 0,00  | 0,00  | 0,01                 | 0,01  | 0,02  | 0,01  | 0,02  | 0,01  | 0,02  | 0,01  |
| $HV$                | 0,05            | 0,05  | 0,07  | 0,07  | 0,09  | 0,04  | 0,07  | 0,06  | 0,10                 | 0,09  | 0,10  | 0,11  | 0,13  | 0,09  | 0,09  | 0,10  |
| $HR(S_I, S_2)$      | 0,02            | 0,02  | 0,02  | 0,02  | 0,01  | 0,01  | 0,02  | 0,02  | 61,21                | 61,59 | 58,12 | 65,52 | 60,39 | 68,53 | 51,79 | 61,02 |
| $Spc$               | 0,03            | 0,03  | 0,06  | 0,05  | 0,06  | 0,02  | 0,04  | 0,04  | 0,05                 | 0,05  | 0,06  | 0,06  | 0,05  | 0,06  | 0,08  | 0,06  |
| $ONVG$              | 14,07           | 11,43 | 13,20 | 12,40 | 12,53 | 9,93  | 14,40 | 12,57 | 20,33                | 15,83 | 18,27 | 18,30 | 24,90 | 15,20 | 13,23 | 18,01 |
| $M_3^*$             | 0,60            | 0,58  | 0,72  | 0,71  | 0,74  | 0,46  | 0,69  | 0,64  | 1,05                 | 1,02  | 1,07  | 1,15  | 1,17  | 1,00  | 0,99  | 1,07  |

**Table 6.5 :** Performance metrics for high demand data set (For binary metrics  $CS$ ,  $D$  and  $HR$ ,  $S_I$  is the pareto optimal set of the related algorithm).

| Performance Measure | PSA based MOGAP |       |       |       |       |       |       |       | eSL-BBBC based MOGAP |       |       |       |       |       |       |       |
|---------------------|-----------------|-------|-------|-------|-------|-------|-------|-------|----------------------|-------|-------|-------|-------|-------|-------|-------|
|                     | Day 1           | Day 2 | Day 3 | Day 4 | Day 5 | Day 6 | Day 7 | Av    | Day 1                | Day 2 | Day 3 | Day 4 | Day 5 | Day 6 | Day 7 | Av    |
| $CS(S_I, S_2)$      | 2,70            | 3,07  | 1,77  | 3,53  | 4,57  | 3,27  | 1,93  | 2,98  | 10,00                | 10,20 | 9,77  | 11,33 | 12,03 | 9,17  | 10,33 | 10,40 |
| $D(S_I, S_2)$       | 0,00            | 0,00  | 0,00  | 0,00  | 0,00  | 0,00  | 0,00  | 0,00  | 0,01                 | 0,03  | 0,02  | 0,02  | 0,03  | 0,02  | 0,02  | 0,02  |
| $HV$                | 0,06            | 0,08  | 0,05  | 0,06  | 0,10  | 0,06  | 0,06  | 0,07  | 0,11                 | 0,17  | 0,11  | 0,12  | 0,14  | 0,11  | 0,11  | 0,12  |
| $HR(S_I, S_2)$      | 0,02            | 0,01  | 0,01  | 0,01  | 0,02  | 0,02  | 0,02  | 0,02  | 59,93                | 78,36 | 74,33 | 63,12 | 49,65 | 58,39 | 67,30 | 64,44 |
| $Spc$               | 0,02            | 0,04  | 0,02  | 0,03  | 0,05  | 0,04  | 0,03  | 0,03  | 0,06                 | 0,05  | 0,06  | 0,06  | 0,05  | 0,07  | 0,07  | 0,06  |
| $ONVG$              | 11,73           | 11,90 | 10,10 | 12,47 | 13,73 | 11,40 | 11,97 | 11,90 | 17,50                | 26,07 | 23,30 | 19,70 | 21,67 | 16,10 | 17,57 | 20,27 |
| $M_3^*$             | 0,54            | 0,63  | 0,48  | 0,61  | 0,79  | 0,57  | 0,56  | 0,60  | 1,08                 | 1,29  | 1,12  | 1,14  | 1,15  | 1,05  | 1,08  | 1,13  |

The pareto front members of both algorithms are normalized with the initial cost score obtained after the GTMA heuristics; and then, the performance metrics are calculated. All the metrics are designed to favor bigger values. Hence, the newly



proposed method clearly outperforms the PSA based MOGAP approach reported in Drexl and Nikulin (2008). Although it is not included in the tables, the first objective function is ameliorated at around ten percent in the proposed method; whereas, the objective score for minimizing the number of planes assigned to apron get worse in PSA based MOGAP.

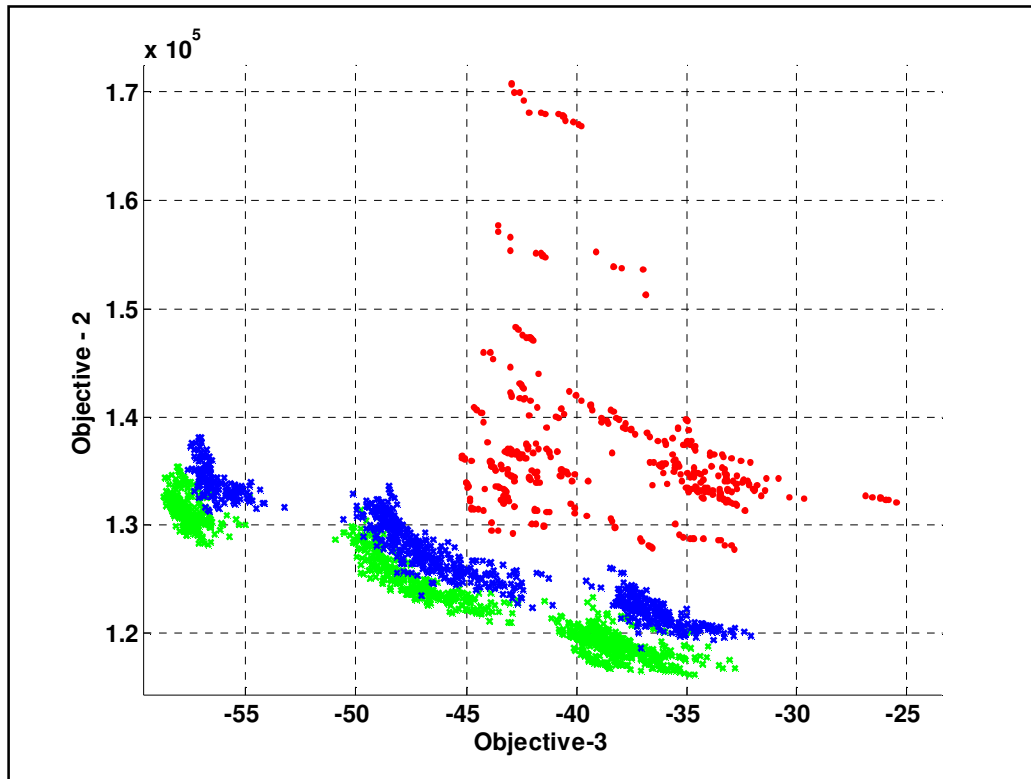
### 6.4.3 Experiments on actual field data

In this part, the experiments are performed on a daily flight schedule data obtained from the operator of the İstanbul Atatürk Airport. There are 359 flights to be assigned to the gates and this corresponds to a demanded time slots / available time slot ratio of 0.71. The performance results for the actual field data are reported in **Table 6.6** for the same testing parameters and performance metrics reported in Chapter 6.4.2.

**Table 6.6** : Performance metric for actual field data. (For binary metrics  $CS$ ,  $D$  and  $HR$ ,  $S_l$  is the pareto optimal set of the related algorithm).

| Performance Measure | PSA based MOGAP | eSL-BBBC based MOGAP |
|---------------------|-----------------|----------------------|
| $CS(S_1, S_2)$      | 0,00000         | 12,66667             |
| $D(S_1, S_2)$       | 0,00014         | 0,08091              |
| $HV$                | 0,10161         | 0,17213              |
| $HR(S_1, S_2)$      | 0,01969         | 53,42133             |
| $Spc$               | 0,04553         | 0,04091              |
| $ONVG$              | 12,66667        | 34,13333             |
| $M_3^*$             | 0,63584         | 1,12716              |

**Figure 6.6** provides an illustration of an experiment on actual field data. In **Figure 6.6**, the pareto fronts of 30 independent algorithm runs for both algorithms are given on the same graph.



**Figure 6.6 :** The complete picture for the 30 independent pareto front representations: points with red dots designate the results of PSA method and blue and green crosses show the results of eSL-BBBC method before and after the archive post processing phase, respectively.

## 6.5 Conclusions

In this chapter, a multi-objective gate assignment problem with the objectives of maximizing gate allocation, minimizing passenger walking distances and maximizing flight to gate preferences is defined and solved.

As the major contributions, a multi-objective nature inspired solution technique; namely, enhanced order based Single Leap-Big Bang Big Crunch (eSL-BBBC) optimization algorithm that possesses the main properties stated below is proposed:

- The result of the algorithm is a set of non-dominated members; this is to say, there are no a priori articulation of the preferences and the algorithm yields a representative set of compromise solutions.
- The optimization method used here is a global evolutionary algorithm.
- The proposed methodology and the generated algorithm have the capacity of handling large data sets, which occur more frequently in real life applications.

- The proposed algorithm uses a plane ordering logic to handle flights and introduces a preference based assignment strategy to assign the flight to a specific gate.

The other main contribution of the study is the implementation of a test data generator for the airport gate assignment problem. The test data generator has the modules for flight data generation, walking distance generation, passenger transfer model generation and flight to gate preference generation.

In order to verify the performance of the algorithm, a set of different metrics has been defined. The two methods are then tested on the sets of artificially generated data as well as actual field data obtained from the İstanbul Atatürk Airport based on these metrics. As it is detailed in the final section, the results obtained through simulations clearly show the effectiveness and the superiority of the newly proposed enhanced order based Single Leap-Big Bang Big Crunch (eSL-BB BC) optimization allocation strategy when compared to the PSA based method.



## 7. CONCLUSIONS AND RECOMMENDATIONS

In this thesis work, a comprehensive study on the extensions and enhancements of the Big Bang-Big Crunch algorithm is given. Then, an increasingly important topic on operations research society, airport gate assignment problem is studied both as a single objective and as a multi-objective problem and it has been tried to be solved using BB-BC based algorithms.

Big Bang-Big Crunch with Local Directional Moves (BBBC-LS) algorithm generates a direction vector from the past positions of the best individuals found so far and investigates on this line with extraction or contraction moves. In addition, well-practiced Nelder-Mead method is presented as a crunching function option. The switching of the crunching function is controlled by a newly introduced switching parameter. The switching threshold parameter is assigned at the beginning of the algorithm and it is kept constant throughout the search. However, it is a promising idea to adapt this parameter in a dynamical manner. One decent idea is to use a feedback controller observing the population diversity and history of the population diversity.

BBBC-LS algorithm is shown to be accurate, effective and fast in the experiments done. Moreover, it is easy to tune the existing parameters within the algorithm.

The second main contribution of this thesis is the new approach proposed for gate assignment operations in the airports. To the best of my knowledge, this thesis work is the first attempt to propose the following:

- I. The binary problem formulation of maximizing total time slots,
- II. Plane order based solution approach for single objective gate assignment problem (SOGAP) : Single Leap -Big Bang Big Crunch (SL-BBBC) Optimization Algorithm,
- III. Plane order based solution approach for multi-objective gate assignment problem (MOGAP) : Enhanced Order Based Single Leap-Big Bang Big Crunch (eSL-BBBC) Optimization Algorithm,

IV. Test data generator engine to provide a quasi-realistic airport plane and pedestrian traffic data.

Airport gate assignment problem is one of the most important issues in operation research as well as in airline industry. Consequently, there is a good opportunity of practically applying academic findings. Using the results of this thesis work, specifically the SL-BBBC algorithm, a resource management system has been constructed in the Atatürk Airport of İstanbul. Performance results collected on the field (Atatürk Airport, İstanbul) are highly correlated with the results on artificially generated test data.

The eSL-BBBC method developed in this work is illustrated to produce a good representation of the pareto optimal set. The next goal is to implement the multi-objective version of the algorithm in the resource management system and visually assisting the operator in his/her decisions.

## REFERENCES

- Akyol A., Yaslan Y. and Erol O. K.** (2007). A Genetic Programming Classifier Design Approach for Cell Images. *Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU*, (Lecture Notes In Artificial Intelligence, Vol. **4724**) 878-888.
- Ahn C. W.** (2006). *Advances in Evolutionary Algorithms*, Springer-Verlag Berlin Heidelberg.
- Ahn C. W., Kim E., Kim H. T., Lim D. H. and An J.** (2010). A Hybrid Multiobjective Evolutionary Algorithm: Striking Balance with Local Search, *Mathematical and Computer Modelling*, vol **52**, pp. 2048-2059.
- Arnold D. V.** (2004). An analysis of evolutionary gradient search. *IEEE Congress on Evolutionary Computation*, **1**:47–54, June.
- Arnold D. V. and MacDonald D.** (2006). Weighted multirecombination evolution strategies on the parabolic ridge. *IEEE Congress on Evolutionary Computation*, pages 104–111.
- Babic, O., Teodorovic, D. and Tasic, V.** (1984). Aircraft stand assignment to minimize walking. *Journal of Transportation Engineering*, **110**(1), pp. 55-66.
- Back T. and Eiben A.E.** (1999). Generalizations of intermediate recombination in evolution strategies. *Proc. of Congress on Evolutionary Computation (CEC'99)* pp. 1566–1573.
- Back T.** (2000). *Self-adaptation*. Chapter 21 of *Evolutionary Computation 2: Advanced Algorithms and Operators*. pp. 188-211, Institute of Physics Publishing, Bristol.
- Baker J.E.** (1987) Reducing bias and inefficiency in the selection algorithm. *Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications*. Lawrence Erlbaum, Hillsdale, New Jersey, pp. 14-21.
- Bambha N. K., Bhattacharyya S. S., Teich J., and Zitzler E.** (2004). Systematic integration of parameterized local search into evolutionary algorithms, *IEEE Trans. Evol. Comput.*, vol. **8**(2), pp. 137–155.
- Banzhaf W., Nordin P., Keller R.E. and Francone F.D.** (1998). *Genetic Programming: An Introduction*. Morgan Kaufmann, San Francisco, 1998.
- Bandara S. and Wirasinghe S.** (1992). Walking distance minimization for airport terminal configurations. *Transportation Research*, **26A**, pp. 59–74.

- Bazargan M.** (2004) *Airline Operations and Scheduling*. Ashgate Publishing Limited, England.
- Beyer H.-G.** (1995). Toward a theory of evolution strategies: On the benefits of sex — the  $(\mu/\mu, \lambda)$ -theory. *Evolutionary Computation*, **3**(1):81–111.
- Beyer H.-G.** (2001). *The Theory of Evolution Strategies*. Springer, Berlin, Heidelberg, New York.
- Bolat A.** (1999). Assigning arriving flights at an airport to the available gates. *Journal of the Operational Research Society*, **50**, pp. 23–34.
- Bolat A.** (2000). Procedures for providing robust gate assignments at airport terminals. *European Journal of the Operational Research*, **50**, pp. 23–34.
- Bolat A.** (2001). Models and a genetic algorithm for static aircraftgate assignment problem. *Journal of the Operational Research Society*, **52**, pp. 1107–1120.
- Camp C.V.** (2007). Design of space trusses using big bang-big crunch optimization. *Journal of Structural Engineering*, **133** (7), 999-1008.
- Chang C.** (1994). *Flight sequencing and gate assignment in airport hubs*. PhD thesis. University of Maryland at College Park.
- Clemente R. M., Puntonet C.G. and Rojas F.** (2003). Post-nonlinear blind source separation using methaheuristics, *Electronics Letters*, 27<sup>th</sup> November Vol. **39** No. 24.
- Coello Coello C. A., Lamont G.B. and Van Veldhuizen D. A.** (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*, Springer.
- Cohon J. L. and Marks D. H.** (1975). *A Review and Evaluation of Multiobjective Programming Techniques*.
- Davis L.** (1991). (Editor) *Handbook of Genetic Algorithms*. Van Nostrand Reinhold.
- De Jong K.A.** (1975). *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan.
- De Jong K.A.** (2006). *Evolutionary Computation: A Unified Approach*. MIT Press, Cambridge, Massachusetts.
- Deb K.** (1999). Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evol Comput* **7**, pp. 205–230.
- Deb K., Agrawal S., Pratab A. and Meyarivan T.** (2000). A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In *M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France. Springer. Lecture Notes in Computer Science No. 1917.
- Deb K., Pratab A., Agrawal S. and Meyarivan T.** (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, **6**(2):182–197, April.



- Deb K., Sinha A., Kukkonen S.** (2006). Multi-objective test problems, linkages, and evolutionary methodologies. *In: Proceedings of the 8th ACM annual conference on genetic and evolutionary computation*, pp 1141–1148.
- Deb K., Thiele L., Laumanns M. et al.** (2001). *Scalable test problems for evolutionary multiobjective optimization*. Tech. rep. 2001001, Kanpur Genetic Algorithms Laboratory. Indian Institute of Technology.
- Ding, H., Lim, A., Rodrigues, B. and Zhu, Y.** (2004). Aircraft and gate scheduling optimization at airports. *Proceedings of the 37th Hawaii IEEE Int. Conf. on System Sciences*, 0-7695-2056-1/04, pp. 1-8.
- Ding, H., Lim, A., Rodrigues, B. and Zhu, Y.** (2004). New Heuristics for over-constrained Flight to Gate Assignments. *Journal of the Operational Research Society*, **55**, pp. 760 – 768.
- Ding, H., Lim, A., Rodrigues, B. and Zhu, Y.** (2005). The over-constrained airport gate assignment problem. *Computers and Operations Research*, **32**, pp. 1867 – 1880.
- Dogan M. and Istefanopulos Y.** (2007). Optimal nonlinear controller design for flexible robot manipulators with adaptive internal model. *IET Control Theory and Applications*, **1** (3), 770-778.
- Dorndorf, U.** (2002). *Project scheduling with time windows: from theory to applications*. Physica-Verlag Heidelberg.
- Dorndorf, U., Drexl, A., Nikulin Y. and Pesch E.** (2007). Flight Gate Scheduling: State-of-the-art and Recent Developments. *The International Journal Of Management Science*, **35**, pp. 326 – 334.
- Drexl, A. and Nikulin, Y.** (2008). Multicriteria airport gate assignment and Pareto simulated annealing. *IIE Transactions* **40**, 385–397.
- Eiben A.E. and Back T.** (1997). Empirical investigation of multiparent recombination operators in evolution strategies, *Evolutionary Computation*, **5**(3):347–365, 1997.
- Eiben A.E. and Smith J.E.** (2003). *Introduction to Evolutionary Computing*. Springer-Verlag, Berlin, Heidelberg, New York.
- El-Mihoub T. A., Hopgood A. A., Nolle L. and Battersby A.** (2004). “Hybrid Genetic Algorithms: A Review”, *Electronic Letters*, vol **13**.
- Erol, O.K. and Eksin, I.** (2006). A new optimization method: Big Bang-Big Crunch. *Advances in Engineering Software*, Elsevier, vol. **37**, pp. 106-111.
- Eshelanian L.J., Caruana R.A. and Schaffer J.D.** (1989). Biases in the crossover landscape. *Proceedings of the 3rd International Conference on Genetic Algorithms*. Morgan Kaufmann, San Francisco, pp. 10-19.
- Fogel L.J., Owens A.J. and Walsh M.J.** (1965). Artificial intelligence through a simulation of evolution. *In: A. Callahan, M. Maxfield, L.J. Fogel, Eds., Biophysics and Cybernetic Systems*. Spartan, Washington DC, pp. 131-156.

- Fogel L.J., Owens A.J. and Walsh M.J.**, (1966). *Artificial Intelligence through Simulated Evolution*. Wiley, Chichester, UK.
- Fogel D.B.** (1998). Ed. *Evolutionary Computation: the Fossil Record*. IEEE Press, Piscataway, NJ.
- Genç H. M.** (2010). A New Solution Approach for the Bearing Only Target Tracking Problem, *IEEE 4th International Workshop on Soft Computing Applications (SOFA'10)*, 15 – 17 July, Arad, Romania.
- Genç H.M., Eksin İ., Erol O.K.** (2010a). Big Bang-Big Crunch Algorithm with Modifications on the Crunching Phase, *International Symposium on INnovations in Intelligent SysTems and Applications, Inista'10*, Kayseri, Turkey.
- Genç H.M., Eksin İ., Erol O.K.** (2010b). Big Bang - Big Crunch Optimization Algorithm Hybridized With Local Directional Moves and Application to Target Motion Analysis Problem, *IEEE International Conference on Systems, Man, and Cybernetics (SMC 2010)*, 10-13 October, İstanbul, Turkey.
- Genç H.M. and Hocaoglu A.K.** (2008). Bearing-only target tracking based on Big Bang - Big Crunch algorithm. *The 3rd Int. Multi-Conference on Computing in the Global Information Technology, (ICCGI 2008)*, 229-233.
- Genç H.M., Erol O.K. and Eksin İ.** (2009). An Application and Solution to Gate Assignment Problem for Atatürk Airport. *DECOMM 2009*.
- Genç, H.M., Erol O.K. and Eksin, İ.** (2011). A Stochastic Meta-Heuristic Approach for the Gate Assignment Problem. *In: Proceedings of the Evolutionary And Deterministic Methods For Design, Optimization And Control (EUROGEN) Conference*, pp. 486-492.
- Genç, H.M., Erol O.K., Eksin, İ., Berber M. F. and Güteryüz B. O.** (2012). A Stochastic Neighborhood Search Approach for Airport Gate Assignment Problem. *Expert Systems with Applications* **39**(1), pp. 316-327.
- Goldberg D.E.** (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.
- Goldberg D.E. and Lingle R.** (1985). Alleles, loci, and the traveling salesman problem. *Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications*. Lawrence Erlbaum, Hillsdale, New Jersey, pp. 154-159.
- Goldberg, D. E., Deb, K., and Clark, J. H.** (1992). "Genetic Algorithms, Noise, and the Sizing of Populations," *Complex Systems*, Vol. **6**, Complex Systems Pub., Inc., , pp. 333-362.
- Gruenz L. and Beyer H.-G.** (1999). Some observations on the interaction of recombination and self-adaptation in evolution strategies. *Proc. of Congress on Evolutionary Computation (CEC'99)* pp. 639–645.
- Gu, Y. and Chung, C.A.** (1999). Genetic algorithm approach to aircraft gate reassignment problem. *Journal of Transportation Engineering*, **125**(5), pp. 384-389.

- Hansen N. and Ostermeier A.** (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, **9**(2):159–195.
- Hansen N., Muller S.D., and Koumoutsakos P.** (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, **11**(1):1–18.
- Holland J.H.** (1973). Genetic algorithms and the optimal allocation of trials. *SIAM Journal of Computing*, **2**, pp.88-105.
- Holland J.H.** (1975). (used edition: 1992) *Adaption in Natural and Artificial Systems*. MIT Press, Cambridge, MA, The University of Michigan Press, Ann Arbor.
- Haghani, A. and Chen, M.C.** (1998). Optimizing gate assignments at airport terminals. *Transportation Research*, **32**(6), pp. 437-454.
- Hansen N., Müller S.D. and Koumoutsakos P.** (2003). Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA–ES), *Evolutionary Computation*, vol. **11**(1), pp. 1–18.
- Hansen N.** (2006). The CMA Evolution Strategy: A Comparing Review, In *J.A. Lozano, P. Larrañaga, I. Inza and E. Bengoetxea (eds.). Towards a new evolutionary computation. Advances in estimation of distribution algorithms*, pp. 75–102.
- Hansen N., Niederberger A.S.P., Guzzella L. and Koumoutsakos P.** (2009). A Method for Handling Uncertainty in Evolutionary Optimization with an Application to Feedback Control of Combustion, *IEEE Transactions on Evolutionary Computation*, vol. **13**(1), pp. 180–197.
- Horstmeier, T. and Haan, F.** (2001). Influence of ground handling on turn round time of new large aircraft. *Aircraft Engineering and Aerospace Technology*, **73**(3), pp. 266-271.
- Hu, X. B. and Paulo, E. D.** (2007). An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. *IEEE Congress on Evolutionary Computation*, 1-4244-1340-0/07, pp. 55-62.
- Huang V.L., Qin A.K. and Deb K. et al.** (2007). *Problem definitions for performance assessment on multi-objective optimization algorithms*. Tech. rep., Nanyang Technological University, Indian Institute of Technology, Swiss Federal Institute of Technology, University Dortmund, The University of Western Australia, Singapore.
- Husband S., Barone L., While L. et al.** (2005). A scalable multi-objective test problem toolkit. In: *Coello Coello CA, Aguirre AH, Zitzler E (eds) Evolutionary multi-criterion optimization*. Springer, Berlin Heidelberg New York, pp 280–295.
- Husband S., Hingston P., Barone L. et al.** (2006). A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans Evol Comput* **10**(5), pp. 477–506.

- Iorio A.W. and Li X.** (2006). Rotated test problems for assessing the performance of multi-objective optimization algorithms. *In: Proceedings of the ACM annual conference on genetic and evolutionary computation*, pp 683–690.
- Kaveh A. and Talatahari S.** (2009). Size optimization of space trusses using Big Bang–Big Crunch algorithm. *Computers and Structures*, **87** (17-18), 1129-1140.
- Knowles J. D. and Corne D. W.** (2000). Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, **8**(2):149–172.
- Koza J.R.** (1992). *Genetic Programming*. MIT Press, Cambridge, MA.
- Koza J.R.** (1994). *Genetic Programming II*. MIT Press, Cambridge, MA.
- Kumbasar T., Yesil E., Eksin I. and Guzelkaya M.** (2008). Inverse fuzzy model control with online adaptation via big bang-big crunch optimization. *3rd International Symposium on Communications, Control, and Signal Processing*, ISCCSP 2008, 697-702.
- Kumbasar T., Eksin I., Guzelkaya M. and Yesil E.** (2008). Big bang big crunch optimization method based fuzzy model inversion, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5317 LNAI, 732-740.
- Li H. and Zhang Q.** (2009). Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. *IEEE Trans Evol Comput* **13**(2), pp. 284–302.
- Lozano M., Herrera F., Krasnogor N. and Molina D.** (2004). “Real-coded memetic algorithms with crossover hill-climbing,” *Evolutionary Computation*, vol. **12**(3), pp. 273–302.
- Mathews J. H. and Fink K.D.** (2004). *Numerical Methods Using Matlab*. Prentice-Hall Inc., Upper Saddle River, New Jersey, USA.
- Matsumura Y., Ohkura K., and Ueda K.** (2001). Evolution Strategies with Multi-Parent Recombination, *Joho Shori Gakkai Shinpojiumu Ronbunshu*, vol 2001/12, pp. 107-114.
- Matsumura Y., Ohkura K., and Ueda K.** (2002). Advantages of Global Discrete Recombination in  $(\mu/\mu, \lambda)$  Evolution Strategies, *Proceedings of the 2002 World on Congress on Computational Intelligence (WCCI)*, Honolulu, HI, USA, 12-17 May, ISBN: 0-7803-7282-4.
- Mendoza N., Chen Y.-W., Nakao Z., and Adachi T.** (2001). "A hybrid optimization method using real-coded multi-parent EA, simplex and simulated annealing with application in the resolution of overlapped signals", *Appl. Soft Comput.*, vol. **1**, pp. 225 – 235.
- Michalewicz, Z.** (1992). *Genetic Algorithms + Data Structure = Evolution Programs*, Springer-Verlag, New York.

- Moscato, P.** (1989). *On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms*. Caltech Concurrent Computation Program (report 826).
- Nelder J. A. and Mead R.** (1965). A simplex method for function minimization. *Computer Journal*, vol 7, pp 308–313.
- Neubauer A.** (1997). Adaptive non-uniform mutation for genetic algorithms. Adaptive non-uniform mutation for genetic algorithms. *Proceedings of the International Conference on Computational Intelligence and Lecture Notes in Computer Science*, Dortmund, Germany. 28.-30. April. Springer-Verlag, Berlin.
- Noman N. and Iba H.** (2008). Accelerating Differential Evolution Using an Adaptive Local Search, *IEEE Trans. On Evolutionary Computation*, vol.12(1), pp.107-125.
- Obata, T.** (1979). *The quadratic assignment problem: Evaluation of exact and heuristic algorithms*. Tech. Report TRS-7901. Rensselaer Polytechnic Institute, Troy, New York.
- Okabe T., Jin Y., Olhofer M. et al.** (2004). On test functions for evolutionary multi-objective optimization. In: *Proceedings of the international conference on parallel problem solving from nature*, pp 792–802.
- Oliver I.M., Smith D.J. and Holland J.** (1987). A study of permutation crossover operators on the travelling salesman problem. *Proceedings of the 2<sup>nd</sup> International Conference on Genetic Algorithms and Their Applications*. Lawrence Erlbaum, Hillsdale, New Jersey. pp. 224-230.
- Ong Y.-S. and Keane A. J.** (2004). Meta-Lamarckian learning in memetic algorithms, *IEEE Trans. Evol. Comput.*, vol. 8(2), pp. 99–110.
- Ong Y.-S., Lim M.-H., Zhu N., and Wong K.-W.** (2006). Classification of adaptive memetic algorithms: A comparative study, *IEEE Trans. Syst., Man, Cybern.—Part B*, vol. 36(1), pp. 141–152.
- Osyczka A.** (1985). Multicriteria optimization for engineering design. In *J. S. Gero, editor, Design Optimization*, pp. 193–227. Academic Press.
- Radcliffe N.** (1991). Forma analysis and random respectful recombination. *Proceedings of the 4<sup>th</sup> International Conference on Genetic Algorithms*. Morgan Kaufmann, San Francisco, pp. 222-229.
- Rechenberg I.** (1973). *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Hozlboog Verlag, Stuttgart.
- Salomon. R.** (1998). Evolutionary algorithms and gradient search: similarities and differences. *IEEE Trans. Evolutionary Computation*, 2(2): 45–55.
- Schott, J.R.** (1995). *Fault tolerant design using single and multicriteria genetic algorithm optimization*. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology.
- Schwefel H.-P.** (1995). *Evolution and Optimum Seeking*. Wiley, New York.

- Schwefel H.-P. and Rudolph G.** (1995). Contemporary evolution strategies. In *F. Mor'an, A. Moreno, J. J. Merelo, and P. Chac'on, editors, Advances in Artificial Life. Third International Conference on Artificial Life, volume 929 of Lecture Notes in Artificial Intelligence*, pages 893–907. Springer, Berlin.
- Spears W.M. and De Jong K.A.** (1991). *An analysis of multi point crossover. Foundations of Genetic Algorithms*. Morgan Kaufmann, San Francisco, pp. 301-315.
- Suganthan P.N., Hansen N., Liang J.J., Deb K., Chen Y.P., Auger A. and Tiwari S.** (2005). *Problem definitions and evaluation criteria for the CEC 2005 Special Session on Real Parameter Optimization*, Tech. Report, Nanyang Technological University.
- Teodorovic D. and Guberinic S.** (1984). Optimal dispatching strategy on an airline network after a schedule perturbation. *European Journal of Operations Research*, **15** pp. 178-182.
- Teodorovic D. and Stojkovic G.** (1990). Model for operational daily airline scheduling. *Transportation Planning and Technology*, **14**, pp.(273–285)
- Wei, D. and Liu, C.** (2007). Optimizing gate assignment at airport based on genetic-Tabu algorithm. *Proceedings of the IEEE Int. conf. on Automation and Logistics. Jinan, China*, pp. 1135-1140.
- Whitley D.** (2000). *Permutations*. Book chapter in *Evolutionary Computation 1: Basic Algorithms and Operators*. Institute of Physics Publishing, Bristol., Chap. 33.3, pp. 274-284.
- Wirasinghe S. and Bandara S.** (1990). Airport gate position estimation for minimum total costs—approximate closed form solution. *Transportation Research*, 24B, pp. 287–297.
- Xu, J. and Bailey, G.** (2001). The airport assignment problem: Mathematical model and a Tabu search algorithm. *Proceedings of the 34th Hawaii IEEE Int. Conf. on System Sciences*. 0-7695-0981-9/01, pp. 1-10.
- Van Veldhuizen D.A.** (1999). *Multiobjective evolutionary algorithms: classifications, analyses, and new Innovations*. Ph.D. thesis, Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB.
- Van Veldhuizen, D.A. and Lamont, G.B.** (1999). Multiobjective evolutionary algorithm test suites. In *J. Carroll, H. Haddad, D. Oppenheim, B. Bryant, and G. B. Lamont, editors, Proceedings of the 1999 ACM Symposium on Applied Computing*, pp. 351–357.
- Yan S. and Huo C.** (2001). Optimization of multiple objective gate assignments. *Transportation Research* 35A, 413–32.
- Yu, X. and Gen M.** (2010). *Introduction to Evolutionary Algorithms*, Springer-Verlag London.

- Zhang Q., Zhou A., Zhaoy S. et al.** (2008). *Multiobjective optimization test instances for the cec 2009 special session and competition*. Tech. rep. CES-487, University of Essex, Nanyang Technological University, Clemson University, Singapore.
- Zhao, X. and Gao X.** (2004). *Evolutionary Programming Based on Non-Uniform Mutation*, MM Research Preprints, pp. 352-374 MMRC, AMSS, Academia Sinica No. 23.
- Zhu, Y., Lim, A. and Rodrigues B.** (2003). Aircraft and Gate Scheduling with Time Windows. *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03)*, 3-5 November, Sacramento, California, USA.
- Zitzler, E.** (1999). *Evolutionary algorithms for multiobjective optimization: methods and applications*. Ph.D. thesis, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland.
- Zitzler E., Deb K. and Thiele L.** (2000). Comparison of multiobjective evolutionary algorithms: empirical results. *Evol Comput* 8(2):pp. 173–195.
- Zitzler E., Laumanns M., and Thiele L.** (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, editors, *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece.

**Url-1** <<http://www.lri.fr/~hansen/>> date retrieved 20.03.2011

**Url-2**<[http://www.georgeevers.org/pso\\_research\\_toolbox.htm](http://www.georgeevers.org/pso_research_toolbox.htm)> date retrieved 15.05.2011

**Url-3**<<http://thetravelinsider.info/airplanetypes.htm>> date retrieved 20.01.2012





## CURRICULUM VITAE

**Name Surname:** Hakkı Murat Genç

**Place and Date of Birth:** Samsun, TURKEY 05.06.1980

**Address:** TÜBİTAK BİLGEM BTE

**E-Mail:** hmuratgenc@gmail.com

**B.Sc.:** Middle East Technical University, Electrics and Electronics Engineering

**M.Sc.:** İstanbul Technical University, Control and Automation Engineering

### **Professional Experience and Rewards:**

Work Experience:

2003 – 2010: Marmara Research Center, Senior Reseracher

2010 – present: Center of Research For Advanced Technologies Of Informatics  
And Information Security, Chief Researcher

Rewards:

1999: METU High Honour List

2002: METU High Honour List

2008: TUBITAK Scientific Publications Incentive Award

### **List of Publications and Patents:**

**Genç H. M.**, Eksin İ., Güzelkaya, M., Yeşil E. "A Rule-base Modification for Time Delay Systems". ASC 2006, The 10th IASTED International Conference on Artificial Intelligence and Soft Computing, 28-30 August 2006, Palma de Mallorca, Spain.

**Genç H. M.**, Cataltepe Z., Pearson T., "A New PCA/ICA Based Feature Selection Method/Yeni Bir Temel/Bağımsız Bileşen Analizi(TBA/BBA) Tabanlı Öznitelik Seçme Yöntemi", IEEE Sinyal İşleme Uygulamaları Konferansı (SİU 2007), 11 – 13 June, Eskisehir, Turkey.

Cataltepe Z., **Genç H. M.**, Pearson T., "A PCA/ICA Based Feature Selection Method and its Application for Corn Fungi Detection", Eusipco (European Signal Processing Conference) 2007, 3-7 September, Poznan, Poland.

Şenyürek L.H., Köksal S., **Genç H.M.**, Aldoğan D., Haklıdır M., "Implementation of Fuzzy Control for Surface Platforms in a Computer Generated Forces Toolkit", International Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA08), 10-12 December, Vienna, Austria.

**Genç H. M.**, Hocoğlu A.K., " Bearing-Only Target Tracking Based On Big Bang – Big Crunch Algorithm", The Third International Multi-Conference on Computing in the Global Information Technology, 27 July – 01 August 2008, Athens, Greece.

**Genç H. M.**, Yesil E., Eksin I., Guzelkaya M., Tekin Ö. A., "A Rule Base Modification Scheme in Fuzzy Logic Controllers for Time-Delay Systems", Expert Systems with Applications, Elsevier, S0957-4174(08)00761-6, 2009 (SCI - A).

**Genç H.M.**, Okutan C., "Modeling, Control and Simulation of a tactical navigation guidance integrated Autonomous Underwater Vehicle (AUV)", Underwater Defence Technology Conference (UDT'09), 9-11 June, Cannes, France.

**Genç, H. M.**, Erol O. K., Eksin İ., "An Application and Solution to Gate Assignment Problem for Atatürk Airport", DECOMM 2009, 26-29 September, Ohrid, Macedonia.

Baştürk T., **Genç H.M.**, Ergüner F., Okutan C., Özkan Ü., "Life Cycle of Weapon Control Systems for Wire Guided Torpedoes", 2010 Spring Simulation Interoperability Workshop International European Multi Conference, 12-16 April, Orlando, Florida, USA.

Okutan C., **Genç H.M.**, "Comparison of Torpedo Guidance Schemes and Modifications on the Trajectory Plan in the Existence of Obstacles/Allied Forces", Defence Technology Conference (UDT'10), 8 – 10 June, Hamburg, Germany.

**Genç H.M.**, Eksin I., Erol O.K., "Big Bang-Big Crunch Algorithm with Modifications on the Crunching Phase", International Symposium on INnovations in Intelligent SysTems and Applications, Inista'10, 21 – 24 June, Kayseri, Turkey.

Okutan C., Ergüner F., Baştürk T., **Genç H.M.**, "Algoritmadan Uygulamaya Modern Atış Kontrol Yzılımı", Savunma Teknolojileri Konferansı (SAVTEK 2010), 23 – 25 June, ODTÜ, Ankara.

**Genç H. M.**, "A New Solution Approach for the Airport Gate Assignment Problem for Handling of Uneven Gate Demands", 12th World Congress on Transportation Research (WCTR 2010), 11-15 July, Lisbon, Portugal.

**Genç H. M.**, "A New Solution Approach for the Bearing Only Target Tracking Problem", IEEE 4th International Workshop on Soft Computing Applications (SOFA'10), 15 – 17 July, Arad, Romania.

**Genç H.M.**, Eksin I., Erol O.K.. "Big Bang - Big Crunch Optimization Algorithm Hybridized With Local Directional Moves and Application to Target Motion Analysis Problem", IEEE International Conference on Systems, Man, and Cybernetics (SMC 2010), 10-13 October, İstanbul, Turkey.

Aruk F., Güven A.F., **Genç H.M.**, Okutan C. C., "Modelling of an Airborne Torpedo Attack and Implementation on a Simulation Environment", Journal of Defense Modeling and Simulation: Applications, Methodology and Technology, published online before print 15, June 2011, DOI: 10.1177/1548512911411331.

**Genç, H. M.**, Okutan C. C., "Torpedo Güdüm Metodlarının Karşılaştırılması ve Durağan / Hareketli Engellere karşı Yörünge Planında Değişiklikler", 4. Ulusal Savunma Uygulamaları Modelleme ve Simülasyon Konferansı (USMOS 2011), 14-15 June, Ankara, Turkey.

**Genç, H. M.**, Erol O. K., Eksin İ., "A Stochastic Metaheuristic Approach for the Gate Assignment Problem", International Conference on Evolutionaryand

Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems (EUROGEN 2011), 14-16 September, Capua, Italy.  
**Genç H.M.**, Eksin I., Erol O.K., “Big Bang - Big Crunch Optimization Algorithm with Local Directional Moves”, Turkish Journal of Electrical Engineering and Computer Sciences (accepted manuscript).  
**Genç, H. M.**, Erol O. K., Eksin İ. Berber M.F. and Güteryüz B.O., “A Stochastic Neighbourhood Search Approach for Airport Gate Assignment Problem”, Expert Systems with Applications, Elsevier, vol 39, pp316 – 327, DOI: 10.1016/j.eswa.2011.07.021. (SCI - A).

#### **PUBLICATIONS/PRESENTATIONS ON THE THESIS**

- **Genç H. M.**, Hocaoglu A.K., 2008: Bearing-Only Target Tracking Based On Big Bang – Big Crunch Algorithm, *The Third International Multi-Conference on Computing in the Global Information Technology*, 27 July – 01 August, Athens, Greece.
- **Genç, H. M.**, Erol O. K. and Eksin İ., 2009: An Application and Solution to Gate Assignment Problem for Atatürk Airport, *DECOMM 2009*, 26-29 September, Ohrid, Macedonia.
- **Genç, H. M.**, 2010: A New Solution Approach for the Bearing Only Target Tracking Problem, IEEE 4th International Workshop on Soft Computing Applications (SOFA'10), 15 – 17 July, Arad, Romania.
- **Genç H.M.**, Eksin I., Erol O.K., 2010: Big Bang - Big Crunch Optimization Algorithm Hybridized With Local Directional Moves and Application to Target Motion Analysis Problem, IEEE International Conference on Systems, Man, and Cybernetics (SMC 2010), 10-13 October, İstanbul, Turkey.
- **Genç, H. M.**, Erol O. K. and Eksin İ., 2011: A Stochastic Metaheuristic Approach for the Gate Assignment Problem”, *International Conference on Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems (EUROGEN 2011)*, 14-16 September, Capua, Italy.
- **Genç, H. M.**, Erol O. K., Eksin İ., Berber M.F. and Güteryüz B.O., 2012: A Stochastic Neighbourhood Search Approach for Airport Gate Assignment Problem. *Expert Systems with Applications*, Elsevier, vol **39**, pp.316 – 327, DOI: 10.1016/j.eswa.2011.07.021.

