Multi-objective Artificial Bee Colony for Multi-objective Quadratic Assignment Problem

Seyedreza Kazemirazi

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the Degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
July 2013
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Elvan Yılmaz
Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Assoc. Prof. Dr. Muhammed Salamah
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Asst. Prof. Dr. Ahmet Ünveren
Supervisor

Examining Committee

1. Asst. Prof. Dr. Adnan Acan
2. Asst. Prof. Dr. Cem Ergün
3. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

Optimization problems are interesting applications in engineering and they are mostly interdisciplinary in nature. This is due to their applications to real life problems. In some real life problems one objective function should be optimized and the aim is detecting the best solution from all possible solutions. These problems are known as Single-objective Optimization Problems (SO). In some other real life problems there is more than one objective function so called Multi-objective Optimization (MO) Problems. In MO Problems the objectives are mostly contradicting with each other. Hence, the aim is finding a class of fittest solutions regarding to all objective functions. Solutions to MO Problems appear in the form of a Pareto-front. The Quadratic Assignment Problem (QAP) is to allocate a set of facilities to a set of locations. There are two issues to consider in QAP. The first is the interaction between facilities which is indicated with a matrix called a flow matrix and the second is the distance between facilities indicated by a distance matrix. There is a new QAP model so called multi-objective Quadratic Assignment Problem (mQAP). In mQAP there are multiple flow matrices but still only one distance matrix. The desired goal of QAP is to assign the facilities to the locations so that, the summation of products between facilities becomes minimal. It follows that, the QAP is a Single-objective Optimization (SO) Problem and the mQAP is a Multi-objective Optimization (MO) Problem. The Artificial Bee Colony (ABC) Algorithm is inspired from honey bees. The ABC is an algorithm basically created to solve the SO Problems. It is a collection of family agents of honey bees that work together to get the job done. There are three kinds of bees and each is responsible for a different job. In this thesis ABC and MOABC have been used for the solution of QAP and mQAP respectively. ABC and MOABC are modified for the solution of QAP and mQAP by using some different crossover and mutation techniques with Tabu Search method. The performance of different updating methods on ABC and MOABC Algorithms is analyzed.

Keywords: Single-objective Optimization (SO), Multi-objective Optimization (MO), Artificial Bee Colony Optimization (ABC), Quadratic Assignment Problem (QAP)

ÖZ

En iyileme problemleri mühendislikte ilginç uygulamalar olmalarına rağmen, doğal olarak farklı disiplinler arası kullanılan problemlerdir. Bunun sebeplerinden biri gerçek yaşam problemlerine uygulanmasıdır. Bazı gerçek yaşam problemlerinde sadece bir amaç fonksiyonu en iyilenmek istenmektedir. Buradaki amaç mevcut problem çözümlerinden en iyisine ulaşmaktır. Bu tip problemlere tek-amaçlı (TA) en iyileme problemleri denilmektedir. Bazı gerçek yaşam problemleri birden çok amaçlı olabilmektedir bunlara çok-amaçlı (ÇA) en iyileme problemleri denilmektedir. ÇA problemlerinde hedefleri çoğunlukla birbiriyle çelişmektedir. Bundan dolayı tüm amaç fonksiyonlarını kullanarak tek çözüm yerine en iyi olan çözümlerden oluşan bir çözüm sınıfı oluşturmaktadır. Bu sınıfa pareto-ön denmektedir. Karesel atama problemi (KAP) bir dizi aracı bir dizi lokasyona verilen lokasyonlar arası uzaklıklar ve araçlar arası akış bilgileri kullanılarak atama yapma problemi olarak tanımlanır. Ayrıca çoklu karesel atama problemlerinde mevcut olup birden çok akış bilgisi kullanılarak yapılan KAP problemleridir. Yapay arı kolonisi (YAK) algoritması gerçek bal arılarından ilham alınarak tek-amaçlı problemleri çözmek için tasarlanmışlardır. YAK algoritması popülasyon tabanlı bir arama algoritması olup sürü zekasına dayalı metasezgisel yöntemlerden birisidir. Algoritma gerçek bal arılarının yiyecek arama davranışlarını modellemeye dayanmaktadır. Bu çalışmada YAK ve MYAK, KAP ve MYAK algoritlamarında KAP ve MKAP problemlerinin çözümü için farklı çaprazlama ve mutasyon teknikleri ile birlikte tabu arama algoritması kullanarak performansları incelenmiştir.

Anahtar Kelimeler: Tek-amaçlı en iyileme, çok-amaçlı en iyileme, yapay arı kolonisi en iyileme, karasal atama problemi.

Dedicated to my parents with love

ACKNOWLEDGMENTS

I would like to say my most profound appreciation to my dear supervisor, Asst. Prof. Dr. Ahmet Ünveren. His knowledge and experience has been a golden key in my research. He did not only teach me how to research, he also taught me how to work as efficiently as possible.

I owe special gratitude to Asst. Prof. Dr. Adnan Acan. I have learned many aspects of research and topics related to the field of my study from him in his lectures and meetings. The meetings he supervised during my Master’s degree together with Dr. Ünveren, were of enormous help for me in gaining knowledge about my field. Also, I desire to express my gratitude to all the other members of the Computer Engineering Department, especially Assoc. Prof. Dr. Ekrem Varoğlu for his special encouragement. He has taught me how to work under time pressure.

I am equally greatly indebted to my lovable parents for their valuable support in all parts of life. I also owe especial thank to my brothers, sisters and other immediate relatives for their stable persuasion.

Last, but not least, I am grateful for all backing from my friends, especially, my great friend Ms. Gunilla Andersson. Her friendship has been and still is a great encouragement to me and has helped me to always aim for high quality work in my academic research. To all the persons I have mentioned I can only say thank you and in appreciation of their support try to do the best as possible in the future.

TABLE OF CONTENTS

ABSTRACT……………………………………………..…….……………..……iii

ÖZ…………………………………………….…………..……………..……...v

DEDICATION…………………………………….………..…………..…...vii

ACKNOWLEDGEMENTS……………………….………….…….…....……viii

LIST OF TABLE………………………………………………………………xii

LIST OF FIGURE……………………………………………………………...xiii

1 INTRODUCTION…………………………………………..………...………1

1.1 Optimization Problems…………………………...………...…..…………1

1.1.1 Multi-objective Optimization Problems……………..……...………3

1.1.2 Pareto-based Algorithms………………..……………..….………...4

1.2 Memetic Algorithm (MA)…………………………………...…..………...8

1.3 Artificial Bee Colony (ABC) Algorithm……………………………..…………9

1.4 Quadratic Assignment Problem…………………….………..……..……...…10

2 THE INSPIRED ALGORITHM FROM HONEY BEE……..……………..……….…..14

2.1 Introduction……………………………………….……...….………......14

2.2 Behavior of Real Honey Bees……………………………………………...15

2.2.1 Food Sources………………..…………………………...…..........15

2.2.2 Employed Foragers………………...…………………………….......16

2.2.3 Dancing…………………………………………….…...……………17

2.2.4 Unemployed Foragers…………………………………….…....…….17

2.3 Inspired Artificial Bee Colony (ABC) Algorithm……………………...........17

2.4 Multi-objective Artificial Bee Colony (MOABC) Algorithm……….………20

2.4.1 Calculating Fitness Function in Multi-objective ABC Algorithm…...21

2.5 Main Steps in Multi-objective Artificial Bee Colony Optimization ……......21

2.6 Important Goals in Multi-objective Artificial Bee Colony Optimization….…22

2.6.1 Generational Distance…………………….……..……………...23

2.6.2 Spread Degree (SD)…………………….…………..……….........23

3 QUADRATIC ASSIGNMENT PROBLEM (QAP)…. ………………...…………24

3.1 Single-objective Quadratic Assignment Problem (QAP)…………...….....24

3.2 Multi-objective Quadratic Assignment Problem (mQAP)……….....……….28

4 PROPOSED ARTIFICIAL BEE COLONY ALGORITHM FOR QAP...…..……….29

4.1 Introduction……………………….…………………….....……......29

4.2 Proposed Artificial Bee Colony (ABC) Algorithm…..………………...……..31

4.2.1 Initialization………...………………...…………………………....34

4.2.2 Employed Phase………………...……...…………………………..34

4.2.3 Onlooker Phase……………...…………...….................................35

4.2.4 Crossover Phase………...…………...……………………………...35

5 ARTIFICIAL MULTI-OBJECTIVE BEE COLONY FOR mQAP…………………..37

5.1 Introduction……………………………….…………………...….......37

5.2 Mutation Operators.……...……………………...…………………………38

5.2.1 Shift Mutation……………………………..…………….……......39

5.2.2 Swap Mutation……………………………..………………..........39

5.4.3 Neighbour Interchange Mutation………………………………..........40

5.4.4 Mixed Mutation…………………………...……………………….40

5.3 Robust Tabu Search (RTS)……………………………………...……………41

5.4 MOABC Algorithm with Different Variations……..…………...……………42

6 EXPERIMENTAL RESULTS....……………………………………...…………..45

6.1 Results for Artificial Bee Colony Algorithm……..…..…...….……………........45

6.2 Results for Multi-objective Artificial Bee Colony Algorithm……..……........47

7 CONCLUSION……...……………………………………………….………….53

REFERENCE……….………….....………………………………..…..………...54

APPENDIX……………………………………………………………………..60

LIST OF TABLES

Table 1: Results of ABC, MABC and CABC Algorithms.……………………………..46

Table 2: Standard Deviations, Best and Worst Results for Instances………........……46

Table 3: Position and Abbreviation of Modified MOABC Algorithms in Figures…..…47

Table 4: Benchmarks and Their Associated Pareto Fronts in Figures…………...……..48

Table 5: Comparison of Obtained Results of Modified MOABC through Metrics..…...49

LIST OF FIGURES

Figure 1: Non-dominated Solutions for Minimization Type of MO Problems…………..5

Figure 2: Pareto-optimal Front and Dominated Solutions……….………..………..…..7

Figure 3: Created Pareto Front by EC Algorithms…..…...…..…………..……………..8

Figure 4: Campus with some Facilities…….…………………..…………..…………...11

Figure 5: Flow and Distance Matrices………………………...……………….……11

Figure 6: Possible Way of Allocating the Resources…………..………………..…..…12

Figure 7: Employed and Unemployed Bees in Search Area……..…………….…..…..…16

Figure 8: Figure of Waggle Dance……………………..…………………………..…...17

Figure 9: Assigning 6 Facilities in 6 Locations……………………………………....25

Figure 10: Permutation for Figure 9……………………………………..…………..25

Figure 11: Placement of Three Facilities to Three Locations……………..……..….....26

Figure 12: Flow and Distance Matrices………………..……………..…………………26

Figure 13: New Placement of Facilities to Locations……………………….…………....27

Figure 14: Calculating the Shipping Cost with New Assignment of Facilities……........27

Figure 15: Possible Way of Swapping the Facilities……….…………..………..……27

Figure 16: Archive Memory which Contains the Fittest Solutions so far…..…..……….29

Figure 17: Standard Form of Crossover Done on Two Parents……………….……......30

Figure 18: Proposed Artificial Bee Colony Algorithm for QAP……..……………..….33

Figure 19: Proposed Crossover Operator……………..……….….……………..…..35

Figure 20: General Form of Mutation…………….………………………………..……38

Figure 21: Shift Mutation………………..……………………………………...39

Figure 22: Swap Mutation……………………..……………………………………..39

Figure 23: Neighbour Interchange Mutation……………………..…………………..40

Figure 24: Crowding Distance……………..…………………..…………..…………42

Figure 25: Multi-objective ABC Combined with RTS………………..……………..44

Figure 29: Pareto Front for KC20-2fl-1uni………………………..……...………….50

Figure 32: Pareto Front for KC30-3fl-1uni………………………...………...…...…..51

Figure 35: Pareto Front for KC50-2fl-1uni…………………...……..………………….52

Figure 26: Pareto Front for KC10-2fl-1uni.……………...……………..………………60

Figure 27: Pareto Front for KC10-2fl-2uni………………...………..………………….60

Figure 28: Pareto Front for KC10-2fl-3uni………………..……..…………………….61

Figure 29: Pareto Front for KC20-2fl-1uni……………………….…………………….61

Figure 30: Pareto Front for KC20-2fl-2uni………..……………...…………………….62

Figure 31: Pareto Front for KC20-2fl-3uni…………..………...……………………….62

Figure 32: Pareto Front for KC30-3fl-1uni…………..………...……………………….63

Figure 33: Pareto Front for KC30-3fl-2uni…………..………...……………………….63

Figure 34: Pareto Front for KC30-3fl-3uni…………..………...……………………….64

Figure 35: Pareto Front for KC50-2fl-1uni……………………….…………………….64

Figure 36: Pareto Front for KC50-2fl-2uni………..……………...…………………….65

Figure 37: Pareto Front for KC50-2fl-3uni…………..………...……………………….65

Figure 38: Pareto Front for KC75-3fl-1uni…………..………...……………………….66

Figure 39: Pareto Front for KC75-3fl-2uni…………..………...……………………….66

Figure 40: Pareto Front for KC10-2fl-1rl…………..………...…………………………67

Figure 41: Pareto Front for KC10-2fl-2rl…………..………...…………………………67

Figure 42: Pareto Front for KC10-2fl-3rl……………..………...………………………68

Figure 43: Pareto Front for KC10-2fl-4rl……………..………...………………………68

Figure 44: Pareto Front for KC10-2fl-5rl……………..………...………………………69

Figure 45: Pareto Front for KC20-2fl-1rl……………..………...………………………69

Figure 46: Pareto Front for KC20-2fl-2rl……………..………...………………………70

Figure 47: Pareto Front for KC20-2fl-3rl……………..………...………………………70

Figure 48: Pareto Front for KC20-2fl-4rl……………..………...………………………71

Figure 49: Pareto Front for KC20-2fl-5rl……………..………...………………………71

Figure 50: Pareto Front for KC30-3fl-1rl……………..………...………………………72

Figure 51: Pareto Front for KC30-3fl-2rl……………..………...………………………72

Figure 52: Pareto Front for KC30-3fl-3rl……………..………...………………………73

Figure 53: Pareto Front for KC50-2fl-1rl……………..………...………………………73

Figure 54: Pareto Front for KC50-2fl-2rl……………..………...………………………74

Figure 55: Pareto Front for KC50-2fl-3rl……………..………...………………………74

Chapter 1

INTRODUCTION

1.1 Optimization Problems

The purpose of this study is to solve the Quadratic Assignment Problem (QAP) (famous problem in optimal sequential allocation) (Koopmans and Beckmann, 1957) by using Artificial Bee Colony (ABC) Algorithm (D. Karaboga, 2005). Optimization problems vary in number of objectives and variable dimension. In a specific case, it is possible to divide optimization problems into two categories. In the first case, the target is to find the single best objective value (Single-objective form) for the given problem. In the next case, the objective is to find points of compromise among two or more satisfactory objective values (Multi-objective form) that should attain simultaneously. Considering the type of objective function in optimization problems, the goal may be to find the maximum value or minimum values of the objective functions.

Optimization problems can be represented mathematically in continual or differentiable function. To deal with these kind of problems there were some single point algorithm, such as gradient decent algorithm (Morse, P. M. and Feshbach, H., 1953), to find the closest local minimum of a function. This algorithm moves from a current point in a direction referring to the negative of the gradient. These sorts of algorithms are able to find solutions very quickly but with a high probability of being stuck at local optima.

In the late 60s and earlier 70s scientists have worked on such algorithms known as Evolutionary Computation (EC) Algorithms (MIT Press, 1993). EC Algorithms are suitable approximation algorithms to deal with optimization problems because of their specific features. These kinds of algorithms are population based and are able to produce a series of solutions in each run instead of a single solution.

Two main subclasses in EC are Evolutionary Algorithms (EAs) (Ashlock, D., 2006) and Swarm Intelligence (SI) Algorithms (presented in the field of cellular robotic systems) (Beni, G., Wang, J., 1989). The EAs are inspired by biological evolution and SI Algorithms are usually inspired from nature. Both EAs and SI Algorithms are known as population-based optimization algorithms. So far, a variety of EAs and SI Algorithms are proposed and their aiming that, individuals cooperate and compete with each other and tend to achieve better and better solutions in the search space. For example, in Particle Swarm Optimization (PSO) Algorithm (Kennedy, J. and Eberhart, R. C.), a particle is simulated according to behavior of birds flocks, and each individual is represented by a position and a velocity. All particles (individuals) aim to get a position through the search space to satisfy the objectives and constraints of the corresponding particle.

In EC Algorithms, individuals (solutions) aim to improve their fitness through a sequence of generations. Regularly, in each iteration (cycle) some updating mechanisms work on the individuals to create new ones. This process (updating mechanism) can be different from one population to the next one or even from one iteration to the other ones. The goal behind the iterations is to converge the solutions' fitness to the best form as soon and as much as possible. EAs and SI Algorithms are also called meta-heuristics algorithms, stochastic and approximation methods.

1.1.1 Multi-objective Optimization Problems

More generally, an optimization problem includes finding the best available values in domain space. It also includes a variety of different types of objective functions. In many cases there are more than one objective functions that should be minimized or maximized simultaneously. These sort of problems are called Multi-objective Optimization (MO) Problems (Coello, 1999). In addition, there are some constraints that should be considered by the problem solver. In some cases, the constraints are equally important and in others there may be some priority in handling the constraints.

As a simple example, suppose you need to attend an important meeting and you have to go to another city or country to do so. Here, you are free to choose the vehicle to do it. According to your situation (budget-wise or time-wise) you should purchase a ticket and satisfy all restrictions (constraints) as much as possible. Should you choose the cheapest ticket or the shortest time? Due to their contradictory nature, these two objectives cannot be linked. Also, their relative importance may vary. There may be a business emergency requiring you go quickly. However, maybe you are on a very tight budget. These are some constraints you have to take into consideration and weigh up as you make your decision, at the same time as you try to avoid any perturbation in your plan to meet the need.

Over the past near decades various meta-heuristic algorithms have been designed for the solution of MO Problems. For example: Aggregative (K.E. Parsopoulos, M. N. Vrahatis, 2002), Pareto-based (C. Lin and Y. Wang, 2007), Sub- population (Maurice C., and James K., 2002), Lexicographic (Carlos A. Coello Coello, et al, 2002) and Hybrid (Zhang, X.-H., et al, 2005) methods have been proposed. The Aggregative methods are designed to work on a single objective. Therefore, the idea behind these algorithms is to map all objectives into one objective. The Pareto-based approaches collect all good (non-dominated) solutions during the iterations. A limited archive keeps these good solutions and it is updated through the execution of algorithms. The Subpopulation approaches divide the population of solution into some sub-populations (the number of sub-populations depends on the number of objectives) and each sub-population optimizes one objective function. Then, all sub-populations combine their solutions, by taking some solutions from each sub-population, to get a trade-off between all objectives. The notion behind the Lexicographic algorithms is that all objectives should be ranked based on the priority of objectives. These algorithms aim to further improve those objectives that are more important. The hybrid approaches attempt to cover the drawbacks by combining two or more algorithms. Thus, they will have extra power and are able to handle the problems by carrying more advantages with merging some algorithms.

1.1.2 Pareto-based Algorithms

Pareto is the name of an Italian economist (Vilfredo Pareto in year 1848–1923), who used the concept of Pareto efficiency or Pareto optimality in his studies. Pareto optimality was defined as a state of economic sequence allocation of resources, and this concept has found a place in engineering applications.

Pareto-optimal (PO) set, in which Pareto optimality is defined in terms of a dominance relation between two solutions is given as follows: given two solutions *U* and *V*, *U*≠*V*, *U* is said to dominate *V* if *U* is not worse than *V* in all objectives and *U* strictly is better than *V* for at least one objective (Adnan Acan and Ahmet Ünveren, 2005). The solution which dominates other solution (s) and where there is no any other solution to dominate it is called a non-dominated solution. A population of non-dominated solutions creates the Pareto-optimal set (Figure 1). In the PO set, all solutions are considered equally important.

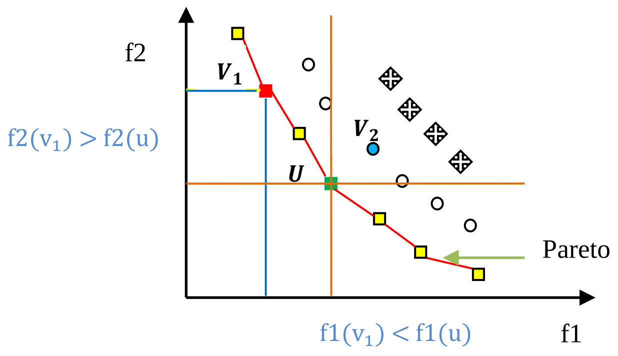


Figure 1: Non-dominate Solutions for a Minimization Type of MO Problems

According to the above definition for dominated and non-dominated solutions, Figure 1 clearly divides the area for dominated and non-dominated solution. As seen in Figure 1, solution U strictly dominates solution V_2. This is because solution U is better than solution V_2 in objectives f_1 and f_2. However, solution U is better than solution V_1 only in objective $f_2$but in comparison with objective $f_1$solution V_1 is better than solution *U*. In this case, solution U cannot dominate solution V_1 and both are non-dominated solutions. In Figure 1, non-dominated solutions are shown with squares and form the PO front.

One main purpose in MO solvers is to extract all non-dominated solutions and preserve them as much as possible. For keeping the non-dominated solutions a limited archive is maintained and the idea is to save and update all non-dominated solutions found so far.

In Pareto-based methods, intensification around the Pareto-optimal front and diversification through the objective space are two main goals. More equally, other guidelines highlight the importance of preserving the non-dominated solutions and work on these as a set of promising solutions. This seems to produce a better result than working on randomly created solutions. Therefore, between all the previously mentioned methods, the Pareto-based approach is very interesting and many researchers have tackled this approach (Greenwald B., 1986).

A general form of minimization problems in multi-objective optimization with constraints (Sanaz Mostaghim and J¨urgen Teich, 2005) can be defined as follows:

$$\min y = f(x) = \left(f_1(x), f_2(x), \dots, f_m(x)\right) \qquad (1)$$

$$\text{subject to } \mathrm{e(x)} = \left(e_1(x), e_2(x), \dots, e_k(x)\right) \leq 0 \qquad (2)$$

$$x \in \mathrm{S}$$

Here, objective functions are shown by $f(x) = \left(f_1(x), f_2(x), \dots, f_m(x)\right)$ and *m* indicates the number of objectives. Generally, the objectives are not commensurable with each other. On the other hand, the minimization should be done considering all objectives at the same time. Hence, it is expected that each solution optimizes all objectives of one given problem. Thus, due to conflicting objectives, it is not possible for solutions to obtain the best values in all objectives simultaneously. In other words, it is expected that if a solution has the best value(s) with regard to one objective, it has worse value(s) in other objective(s). Hence, the result should be a trade-off taking all objectives.

According to the explanation in the previous section about the dominance relation, considering two solutions $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$, it can be explained as follows:

Here, $\boldsymbol{x_1 \epsilon S}$ is a decision vector and it can dominate another decision vector $\boldsymbol{x_2 \epsilon S}$ if both following constraints are met for at least one objective i= 1, …, *m*.

1: x_1 is not worst than x_2 with respect to all objectives, i.e.:

$$f_i(x_1) \leq f_i(x_2) \tag{3}$$

2: x_1 is strictly better than x_2, at least in one objective, i.e.:

$$f_i(x_1) < f_i(x_2), \tag{4}$$

Finally, a decision vector x_1 called Pareto-optimal front (Figure 2) if there exists no other decision vector x_2 to dominate it (K. Deb, et al, 2000).

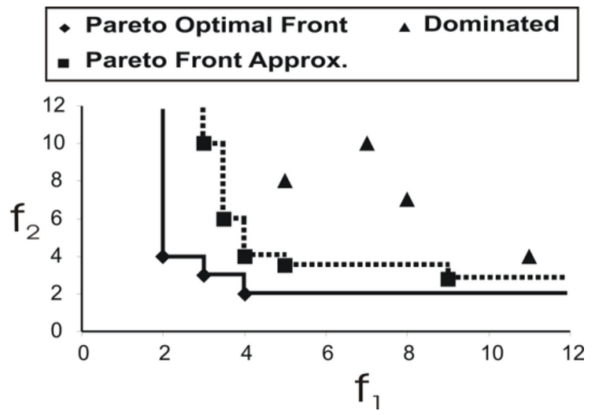


Figure 2: Pareto-optimal Front and Dominated Solutions

Since EC Algorithms are population-based approaches, a series of fittest solutions can be provided per a run by these algorithms (Figure 3). Hence, they are capable to create one Pareto front in each run and through the iteration get closer to Pareto-optimal set and also spread the solutions through the objective space.

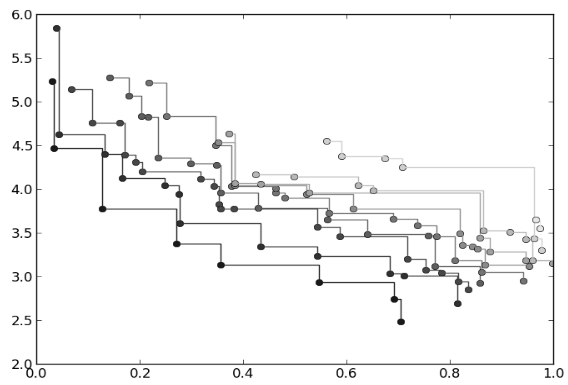


Figure 3: Created Pareto Front by EC Algorithms

1.2 Memetic Algorithms (MA)

Memetic Algorithms (MAs) appeared in the late 80s (Moscato, P., 1989). The main goal in MAs is to cover the inability of heuristics and meta-heuristics to deal with optimization problems. The general idea behind MAs is to have extra power in their search mechanism in all aspects. To this point, MAs are a blending of more than one algorithm to get extra power. Initially, MAs were introduced as stochastic global searches for solving specific problems. However, lately they have been used for local search. For example, a mixture of an evolutionary algorithm, like Genetic Algorithm together with a local searcher can be named a MA. This has been suggested by Simulating Annealing (SA) Algorithm (Kirkpatrick, S., Gelatt Jr., C., and Vecchi) or Tabu Search (TS) (Glover, F., and Laguna, M.), etc. In this kind of MAs individuals improve along with a collaboration of mechanisms (in both local and global aspects).

Earlier the MAs were not recognized as different algorithms because they appeared to be partially equal to EAs. It was therefore hard for scientists to accept this approach as a new mechanism. However, nowadays the MA has become a popular algorithm even in handling hard real problems. In MA the emphasis is on extracting all available knowledge which most probably has not been completely realized in simple EAs. This process of extracting can be done with combining heuristics, local search algorithms, special updating mechanisms, approximation, exact method, etc. It should be mentioned that MA has been known by different names such as Hybrid EAs, Lamarckian EAs, etc.

1.3 Artificial Bee Colony (ABC) Algorithm

Lately, Artificial Bee Colony (ABC) Algorithm as a kind of SI approach is proposed for optimization problems (D. Karaboga, 2005). In general, it is simulated by the process of finding flowers for extract (sources) by the real honey bees in nature. In detail, the ABC Algorithm is simulated based on their specific characteristics such as food foragers, special dance (waggle dance), selection mechanism, routing, social decision, etc. The idea behind the honey bees foraging has attracted special interest form scientists broadly in solving hard optimization problems. This approach includes: self-organizing, dynamic decision in labor to forage and reaching maximum efficiency. The division of labor is leaded out by different sorts of bees. There are four essential properties in ABC that govern the bees namely positive feedback, negative feedback, fluctuation and social interaction done with waggle dance, food source exhausted, random selection and group decision respectively. Accordingly, the proposed ABC algorithm by Karaboga is able to solve combinatorial optimization problems such as traveling salesman, job shop, scheduling and resource allocation. The ABC Algorithm has focused on balancing the exploration and exploitation in search space in defined problems. The exploration is done in the primary phase of search and the exploitation in a later phase of search in optimization problems. However, ABC has good enough power in some parts and weakness in some other. This study has focused on removing the imperfections as much as possible.

The ABC Algorithm was essentially designed to resolve the single objective optimization problems but after a while scientists introduced variations of ABC. Recently a Multi-objective ABC Optimization (MOBCO) Algorithm has been suggested to handle the MO Problems (R. Hedayatzadeh, et al, 2010).

Since the MOABC Algorithm is a kind of SI Algorithm, it is population based and able to produce a series of solutions in each run. Hence, it is a promising algorithm in that it creates a simple Pareto front per a run and is capable of going quickly and get closer toward to the Pareto-optimum front. The MOABC Algorithm with a collective intelligence behavior of honey bees enables the approach to easily deal with MO Problems in cases where the SO Algorithms are not effective solvers.

1.4 Quadratic Assignment Problem (QAP)

Quadratic Assignment Problem (QAP) is an interesting famous combinatorial optimization problem (Koopmans and Beckmann, 1957) and has attracted research by many scientists. It is used to model different real life problems in different areas such as facilities location, parallel and distributed computing and combinatorial data analysis

(Loiola et al., 2005). As a simple example for QAP, consider a construction of a campus (Figure 4).

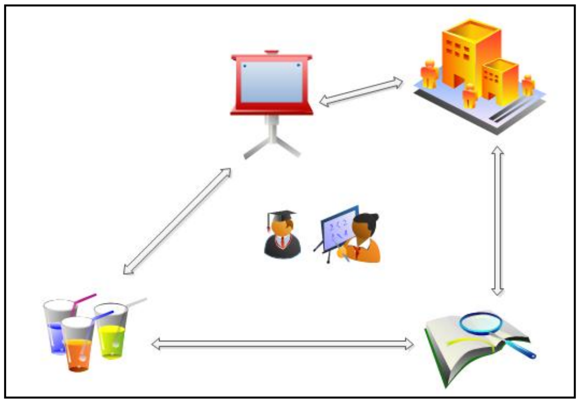


Figure 4: a Campus with Some Facilities

Some buildings with facilities would need to be built in some predefined locations. The point is that the distance between structures should be the least for staff, teachers and students who want to walk between them during the day. Ordinarily, the number of facilities is the same as the number of locations. Hence, each facility should be placed in only one location.

In QAP there are two matrices entitled flow matrix and distance matrix (Figure 5). The flow matrix keeps the relation between facilities and the distance matrix keeps the distance between facilities.

$$F = \begin{bmatrix} 0 & 4 & 3 & 5 \\ 4 & 0 & 3 & 2 \\ 3 & 3 & 0 & 2 \\ 5 & 2 & 2 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 & 1 & 3 & 2 \\ 1 & 0 & 5 & 3 \\ 3 & 5 & 0 & 4 \\ 2 & 3 & 4 & 0 \end{bmatrix}$$

Figure 5: Flow and Distance Matrices

According to the flow matrix in Figure 5, the desired allocation of facilities to predefined location can be done as it comes in Figure 6.

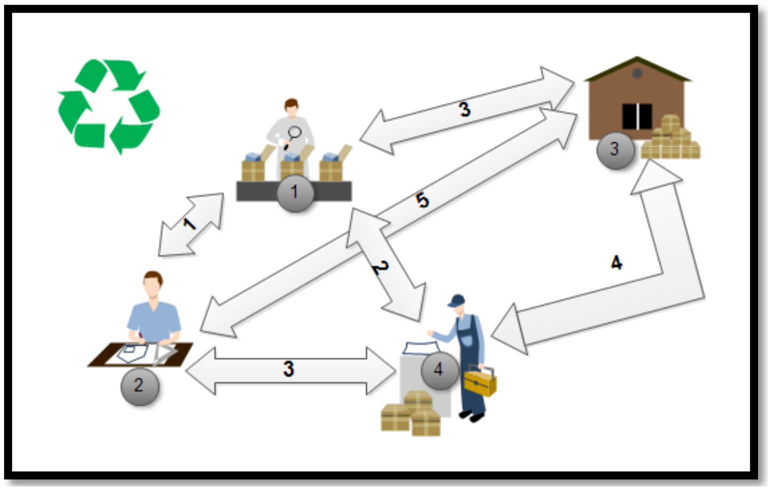


Figure 6: Possible Way to Allocating the Resources

Here, the facilities have shown with the numbered circles behind each. The distance between the locations is also mentioned with the numbered flows.

Since, the subject is to allocate all facilities to the available locations aiming of minimizing the summation of the distances multiplied by the related flows, facilities with more relation with each other should be located closer to each other. For example, facility number 1 has a high amount of relation with facility number 2 and clearly to have a shipping cost as low as possible, these facilities should be allocated near each other. Also the amount of relation between facilities number 2 and 3 is not high as other and therefore these two facilities are considered far away to each other.

Knowles and Corn (Knowles, J. D. and Corne, D.W., 2002), proposed a variation of QAP so called Multi-objective Quadratic Assignment Problem (mQAP). They have presented the mQAP as having multiple flow matrices but still only one distance matrix.

In mQAP, all objectives are a simple standard QAP individually and they use their own flow matrix. Therefore, assigning well-located facilities with respect to only one objective (with regard to one flow matrix) may lead these facilities to have a poor placement with respect to other objective(s). Hence, having a trade-off between all objectives is a definite need.

In the rest of this study the emphasis is to show and discuss in more detail the topics in this introductory chapter. Chapter 2 compromises an introduction of the Bee Colony in nature and discusses the defined algorithm based on this procedure. Chapter 3 is focused on our case study of the Quadratic Assignment Problem (QAP). It highlights two sides of this problem, namely the problem of a single objective QAP and the problem of multi-objective QAP (mQAP). Chapters 4 and 5 have been devoted to the new algorithms and ideas for single and multi-objective ABC Algorithms. The results of our research are set out in chapter 6. This chapter contains comparative results of our work with some other algorithms and clearly compares the results with respect to some standard metrics. Finally, chapter 7 is dedicated to conclusions and a summary of all research which is done in this area.

Chapter 2

INSPIRED ARTIFICIAL BEE COLONY ALGORITHM FROM HONEY BEES

2.1 Introduction

Artificial Bee Colony (ABC) Algorithm has been extracted from the honey bees' working process and the model according to which they arrange themselves (T.D., Seeley). Then, Dušan Teodorović advanced the algorithm further by introducing the Artificial Bee Colony Optimization (ABCO) Algorithm in 2005 (Teodorovic and Orco, 2005). Recently, ABC algorithm is used broadly under scientists' consideration more and more. Dervis Karaboga in 2005 improved the ABC Algorithm by using it especially in numerical optimization (D. Karaboga, 2005) for solving hard optimization problems.

The Multi-objective Artificial Bee Colony Optimization (MOABC) Algorithm is an extension of the original ABC Algorithm aiming at finding a set of optimal solutions for Multi-objective Optimization Problems (Malcolm and Chwee, 2009). The MOABC Algorithm can find multiple Pareto-optimal solutions in a single run (Figure 3).

The first half part of this chapter is devoted to the honey bees' foraging routine with regard to how the plenty of jobs is divided between the different bees. The second half is dedicated to the optimization algorithm inspired from this process.

2.2 Behavior of Real Honey Bees

Karlvon Frisch is a famous Austrian ethologist who received the Nobel Prize. He has been focused on investigation of the sensory perceptions of the honey bees in his works. He found that there are some variety of information in their waggle dance communications and special odor. Through this information, honey bees are able to have a self-organization and are easily capable to find out food sources with high amount of nectar. To have a self-organization, some components are necessary such as: food sources, employed and unemployed foragers and a specific dance. These components have been observed inside and outside the hive. The following is considered to explain these components in detail.

2.2.1 Food Sources

The main goals in ABC are food sources (flowers) (*A* and *B* in Figure 7). The bee can extend itself so far and in different directions to draw out a food source. Principally, flowers with more nectar are visited by more bees, while flowers with less nectar attract fewer bees. Furthermore, each food has different identification with respect to its coordinate, distance from hive, etc. The sort of bees is separated into three distinct species and each is in charge for its specific job.

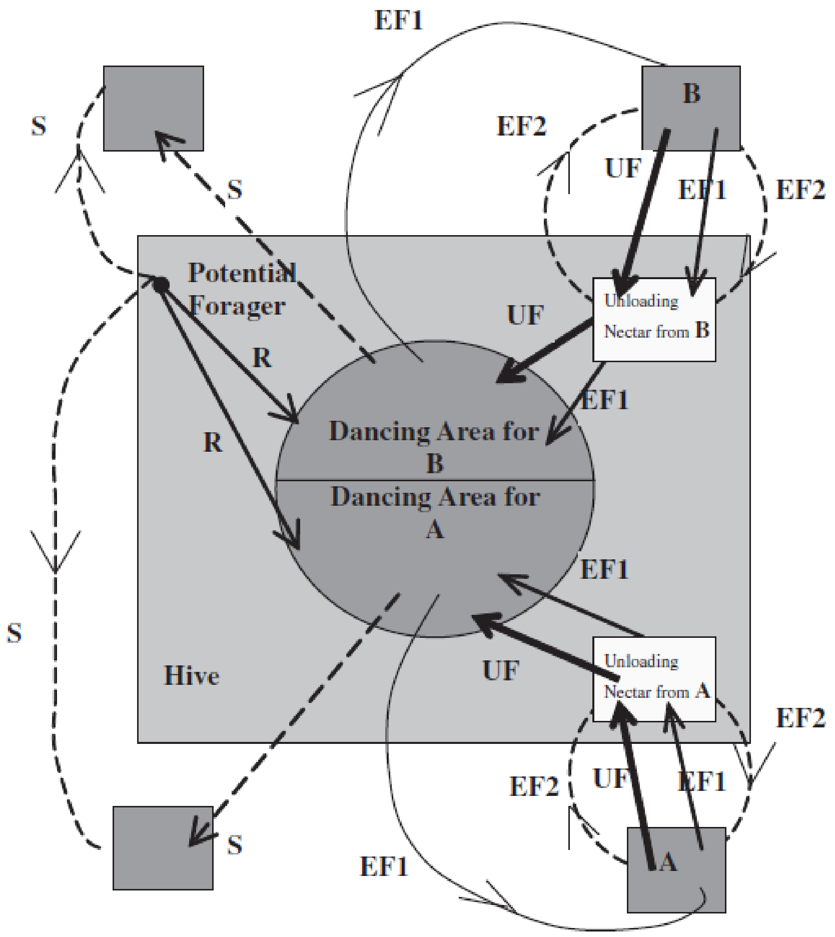


Figure 7: Employed and Unemployed Bees in Search Area
[Chunfan Xu, Haibin Duan]

2.2.2 Employed Foragers

Employed foragers (*EF*1 and *EF*2 in Figure 7) are linked with food sources which are already exploited. They take sufficient information about the food source like profitability, coordination, etc. Then they return to their hives and share the information with all the other bees. This information is transferred by the waggle dance.

2.2.3 Dancing

In each hive there is a section so called dancing space where employed bees transfer their information about food source through the waggle dance. This type of bees' dance is in the shape of an eight (8) figure. It includes crawl, turn and moving around. There are angles, axis and weight in directions. This angular dance shows the food direction with respect to the sun as shown in Figure 8.

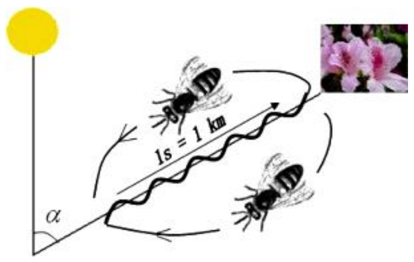


Figure 8: Figure of Waggle Dance [Chunfan Xu, Haibin Duan]

2.2.4 Unemployed Foragers

Unemployed forager bees (*UF* in Figure 7) are those that are looking for in the hives to get the information about the food source through the employed forager waggles' dance. With this information they are able to find the related food source and exploit it. The selection process is according to the profitability of food sources. Obviously food sources with high amount of nectar will attract more bees. Unemployed foragers are divided to two types of bees such as scouts (*S* in Figure 7) and onlookers (*R* in Figure 7).

2.3 Inspired Artificial Bee Colony (ABC) Algorithm

The behavior of real honey bees adapted to the optimization problem solution by defining ABC Algorithm. The ABC Algorithm is divided to four phases namely: initialization, employed, onlooker, and scout phases. Firstly, the initialization phase creates a population of solutions so called X_i, randomly. Then, X_i solutions are sent to a loop to improve through some updating mechanisms. The loop includes employed, onlooker, and scout phases. In employed and onlooker phases, the solutions are disturbed to create new solution (V_i and U_i) and get better solutions than the current one. Also, the scout phase is considered to create a new random solution (X_i) instead of abandoned solution which did not have a satisfactory improvement. Here, there is a pseudo code for ABC Algorithm. Note that, in this algorithm, in each phase some equations are used which are referenced and they are explained in the next page.

Main Steps in ABC Algorithm

```
Initialize the population of solution $X_i$, randomly by Eq. 5;
Evaluate the population (calculate the fitness of each solution);
Cycle=1;
Repeat:
  Employed Phase:
     Generate candidate solution $V_i$ from $X_i$ by Eq. 6;
     Evaluate solution $V_i$;
     Replace $V_i$ instead of $X_i$ if its fitness is better; otherwise keep $X_i$;
     Calculate probability values $P_i$ for all solutions so far by Eq. 7;
   Onlooker Phase:
     Generate candidate solutions $U_i$ from $X_i$ by Eq. 8;
     Evaluate solution $U_i$;
     Replace $U_i$ instead of $X_i$ if its fitness is better; otherwise keep $X_i$;
   Scout Phase:
     If there is an abandoned solution, produce a new solution $X_i$ by Eq. 9;
  Maintain the best solution attained as yet;
  Cycle=Cycle+1;
Until a termination condition is met.
```

The equations mentioned in the above pseudo code are explained below:

$$X_{ij} = X_{j\,min} + rand(0,1)\left(X_{j\,max} - X_{j\,min}\right) \qquad (5)$$

Solutions created at the first stage may appear to have a possible violation which happens when a feasible solution violates the default boundary constraint. To this point, $X_{j\,min}$ and $X_{j\,max}$ are used in Eq. 5 to prevent this violation. It is noticeable that, in the initialization phase, scout (foragers) bees are responsible to find the unseen food sources. Because of this, creating the solution is completely random.

$$V_{ij} = X_{ij} + \phi_{ij}\left(X_{ij} - X_{kj}\right) \qquad (6)$$

Here, $k \in \{1,2,\dots,Ns\}$ and Ns is the total number of food sources which is the same as X' index. Also, $j \in \{1,2,\dots,n\}$ and n shows the dimension of the problem. Note that, all mentioned variables are randomly selected. Implicitly, k must be different from i. The ϕ_{ij} is a random number between the range [-1, 1].

At the end of employed phase, based on solutions' fitness, a probability (P_i) value is assigned to each solution which is a proportion of solutions' goodness. In this case, solution with the highest fitness in the population gets the highest probability.

$$P_i = \frac{f(X_i)}{\sum_{i=1}^{Ns} f(X_i)} \qquad (7)$$

Here, $f(X_i)$ is the fitness value of solution i and Ns is the total number of food sources. Note that, the number of food sources is the same as the number of employed and unemployed (onlooker) bees.

$$U_{ij} = X_{ij} + \phi_{ij}\left(X_{ij} - X_{kj}\right) \qquad (8)$$

All variables are the same as Eq. 6 and the only difference in this phase is in selection mechanism. In employed phase, the modification is done on all solutions in population respectively; however, here the selection is based on the computed probability in the employed phase. Hence, the highest amount of probability has more chance to be selected.

$$X_{ij} = X_{j\,min} + rand(0,1)\big(X_{j\,max} - X_{j\,min}\big) \qquad (9)$$

To prevent of any perturbation, in the variable X_{ij} a lower and upper bounds are considered by $X_{j\,min}$ and $X_{j\,max}$ respectively. When a food source could not get optimized in a few cycles, a scout bee will do this random search on the problem space to find a new food source.

2.4 Multi-objective Artificial Bee Colony Algorithm

In the Multi-objective Artificial Bee Colony (MOABC) Algorithm, same as the Single-objective ABC, food sources represent possible solutions. For example, solution X_i= *(i=1, 2,..., N)* in which, X_i= $(x_{i1}, x_{i2}, x_{i3}, \dots, x_{ij}, .., x_{iD})$ the so called decision variable in which D shows the dimension of the problem. The goodness of the solution X_i, measured by more than one fitness functional and it can be shown by fitness value set *Q*= $\{f_1(X_i),\ f_2(X_i), \dots, f_m(X_i)\}$ where, m is the number of objectives. The number of food source is equal to the number of employed bees. Hence, each food source will attract only one employed bee. The idea behind MOABC is to preserve a set of solution instead of a single solution. It also required to converge to the Pareto-optimal front and maintains a good distribution of solutions through it. For this reason, in dealing with the real value problem, both non-dominate ranks (Omar Al Jadaan et al, 2008) and crowding distance (Carlo R. et al, 2005) are two important issues which may use in MOABC Algorithm for calculating the fitness function. However, these techniques are not used in the MOABC for the solution of mQAP in calculating the fitness function. Only crowding distance is that used in the last part of MOABC for eliminating the crowding solutions (more detailed information is given in chapter 5).

2.4.1 Calculating Fitness Function in Multi-objective ABC Algorithm

In MOABC Algorithm for the solution of real value problems, for calculating the $f_m(X_i)$, both Crowding Distance (CD) and the amount of dominated solutions (non-dominate ranks) by solution X_i are required. The related equation can be defined as following:

$$f(X_i) = \frac{1}{1 + r(X_i) + d(X_i)} \qquad (10)$$

Here, $r(X_i)$ is the number of solutions dominated by X_i, and $d(X_i)$ is the Crowding Distance of X_i.

It is noticeable that, this study did note use this equation for measuring the fitness function in mQAP. The calculating of fitness function in mQAP requires some other techniques which are explained further.

2.5 Main Steps in Multi-objective ABC Algorithm

In the MOABC Algorithm, still there are four phases namely: initialization, employed, onlooker, and scout phases. The MOABC is a Pareto based algorithm with an external archive to store non-dominated solutions. For preserving the latest solutions found so far and the solutions in the archive, solutions should be checked if they dominate each other or not.

To this point, created solutions in the initialization phase, are saved in set S_0 to use in the main loop of the algorithm. Also, created solutions in the employed, onlooker, and scout phases are sent to set C_{t}. Then, set $S_{\mathrm{t+1}}$ is created by combining the set S_{t} and C_{t}. Lastly, next generation is selected from this combination and modification is done on this set. The MOABC's pseudo code is given below.

```
01: Cycle=0, set $S_0$;
02: Initialize parameters;
03: Initialize the initial random solutions $X_i$ by using Eq. 5 and add them to the set $S_t$;
04: Calculating the fitness functions for new solutions;
05: Chose non-dominated solution from $S_t$ and send them to archive;
06: Repeat,
07:     Employed Phase,
08:         Create new solution $V_i$ from $X_i$ by using Eq. 6;
09:         Calculating the fitness functions for new solutions;
10:         Add the new solution to the set $C_t$;
11:     Onlooker Phase,
12:         Select a solution $X_i$, and generate a new solution $U_i$ by using Eq. 8;
13:         Calculating the fitness functions for new solutions;
14:         Add the new solution to the set $C_t$;
15:     Scout Phase,
16:         For abandon solution, generate a new random solution $X_i$ by using Eq. 9;
17:         Add the new solution to the set $C_t$;
18: Update the non-dominated solution in archive by the solutions in $C_t$;
19: Create the set $S_{t+1}$ from the union of set $S_t$ and set $C_t$;
20: Cycle=Cycle+1;
21: Until the stopping criteria are met.
```

2.6 Most Important Goals in Multi-objective ABC Algorithm

Divergence and convergence through the Pareto-optimal front are two goals in Multi-objective Optimization and also the main concern of the MOABC Algorithm. These two goals can be controlled by Generational Distance and Spread degree as they are explained below.

2.6.1 Generational Distance (GD)

Generational Distance (GD) measures the closeness of a particular solution and its closest solution in the Pareto-optimal set. The GD's formula is considered in the following:

$$\mathrm{GD} = \frac{\sqrt{\sum_{i=1}^{N} d_i^2}}{n} \qquad (11)$$

Here, d_i counts the Euclidean Distance of each solution with its closest point in the Pareto-optimal set. Obviously, if the GD is zero there is no distance between current solutions and the Pareto-optimal solution set and all points are in the best situation.

2.6.2 Spread Degree (SD)

The SD is the measurement for deploying a scale of solutions in Pareto-optimal region. Having a set of solutions with a sufficient SD through the Pareto-optimal region is a goal in MOABC Algorithm. The equation below shows the SD expression:

$$\mathrm{SD} = \frac{\sum_{i=1}^{m} d(E_i,S) \sum_{i=1}^{N} |d(X_i,S) - \bar{d}|}{\sum_{i=1}^{m} d(E_i,S) + |S|\bar{d}} \qquad (12)$$

Where, E_i'*s= (i =1, 2, ..., m)* contains *m* excessive solutions in the area of the Pareto-optimal front. The $d(E_i, S)$ measures the Euclidean Distance in the objective space among the E_i and the closest point in S. Finally $d(X_i, S)$ calculates the Euclidean Distance among the solution X_i and the closest point to it in S.

$$\bar{d} = \sum_{i=1}^{N} d = (X_i, S)/|S| \qquad (13)$$

The value of *SD* becomes zero if all solutions in *S* are well done distributed among the objective space and it includes the solutions in E_i's space.

Chapter 3

QUADRATIC ASSIGNMENT PROBLEM (QAP)

3.1 Single-objective Quadratic Assignment Problem (QAP)

Quadratic Assignment Problem (QAP) is one of the challenging classical combinatorial optimization problems. QAP was presented by Koopman and Beckman in 1957 (Koopmans and Beckmann, 1957). It is a model for many practical problems like backboard wiring, campus and hospital layout, and scheduling (Adnan Acan and Ahmet Ünveren, 2005).

To have an instance of QAP, the full list of distances among available locations (d_{rs}) and material flow among facilities (F_{ij}) and should be visible. There are N facilities and each of them can be interchanged with each other. On the other hand, there are N locations each of them can be provided for only one facility. Hence, the QAP can be modeled as follows:

There are a set of N facilities and a set of N locations. For eachpair of locations, a distance (d_{ij}) is specified and for each pair of facilities a flow (F_{ij}) is specified. The problem is to assign all facilities to different locations with the goal of minimizing the sum of the distances multiplied by the corresponding flows [Koopman and Beckman, 1957]. Formally, let d_{ij} and F_{ij} be two N*N matrices and let P_n be the set of permutation of {1, 2, …, N}.

Then, the QAP can be defined as follows:

$$\min \quad f(\varphi) \quad = \left[\sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} . f_{\varphi(i),\varphi(j)} \right] \qquad (14)$$

Over all permutations φ ϵ P_n [Koopman and Beckman, 1957].

Figure 9 and Figure 10 shows a map that 6 different facilities located on 6 different locations.

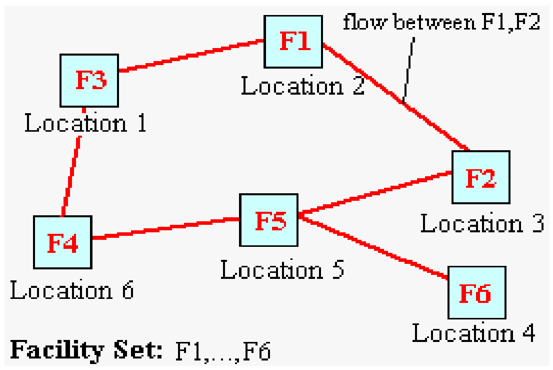


Figure 9: Assigning 6 Facilities to 6 Locations

Here, is a permutation that gives formal definition of Figure 9.

Facilities $\longrightarrow$	6	2	3	5	4	1
Locations $\longrightarrow$	1	2	3	4	5	6

Figure 10: Permutation for Figure 9

According to the allocation in Figure 10, facility 5 has been assigned to location 4, facility 6 to location 1 and so on. The fitness function (*f*) of this example can be calculated as follows:

$$F = \left[\sum_{i=1}^{6} \sum_{j=1}^{6} d_{ij} . f_{\varphi(i),\varphi(j)} \right]$$

Suppose the following example, for three facilities and three locations.

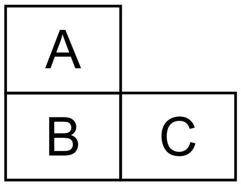


Figure 11: Placement of Three Facilities to Three Locations

The goal is to find a placement of facilities to location to get the fitness value as low as possible with respect to distance and flow matrices (Figure 12).

$$D = \begin{array}{c|ccc|} & A & B & C \\ A & 0 & 1 & 2 \\ B & 1 & 0 & 1 \\ C & 3 & 1 & 0 \end{array} \qquad F = \begin{array}{c|ccc|} & 1 & 2 & 3 \\ 1 & 0 & 1 & 1 \\ 2 & 2 & 0 & 2 \\ 3 & 3 & 1 & 0 \end{array}$$

Figure 12: Flow and Distance Matrices

A possible assignment of facilities to locations can be considered as follow:

$1 \rightarrow$ A, $2 \rightarrow$ B, $3 \rightarrow$ C, i.e. $x_{1A} = 1$, $x_{2B} = 1$, $x_{3C} = 1$, all other $x_{ij} = 0$.

Total cost: 0*0 + 1*1 + 2*1 + 1*2 + 0*0 + 1*2 + 3*3 + 1*1 + 0*0 = **17.**

The solution is not desired whereof, the facility number 1 and 3 (with a high value of material flow) are allocated to locations *A* and *C* (which have the most distance among them). To improve the solution a new assignment is considerd as follow: $1 \rightarrow$ C, $2 \rightarrow$ A and $3 \rightarrow$ B, i.e. $x_{1C} = 1$, $x_{2A} = 1$, $x_{3B} = 1$.

Hence, the distance matrix should be resorted as follow:

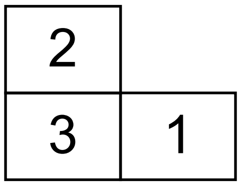


Figure 13: New placement of Facilities to Locations

$$D = \begin{array}{c|ccc} & A & B & C \\ \hline A & 0 & 1 & 2 \\ B & 1 & 0 & 1 \\ C & 3 & 1 & 0 \end{array} \dashrightarrow D = \begin{array}{c|ccc} & C & A & B \\ \hline C & 0 & 3 & 1 \\ A & 2 & 0 & 1 \\ B & 1 & 1 & 0 \end{array}$$

$$D = \begin{array}{c|ccc} & C & A & B \\ \hline C & 0 & 3 & 1 \\ A & 2 & 0 & 1 \\ B & 1 & 1 & 0 \end{array} \quad \text{Multiply,} \quad F = \begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 1 & 1 \\ 2 & 2 & 0 & 2 \\ 3 & 3 & 1 & 0 \end{array}$$

Figure 14: Calculating the Shipping Cost for New Assignment of Facilities

Such that row and columns appear the following sequence:

1 $\rightarrow$ C, 2 $\rightarrow$ A and 3 $\rightarrow$ B, i.e C, A, B

$$D = \begin{array}{c|ccc} & A & B & C \\ \hline A & 0 & 1 & \boxed{2} \\ B & 1 & 0 & \boxed{1} \\ C & 3 & 1 & \boxed{0} \end{array} \Rightarrow D = \begin{array}{c|ccc} & C & A & B \\ \hline A & 2 & 0 & 1 \\ B & 1 & 1 & 0 \\ C & \boxed{0} & \boxed{3} & \boxed{1} \end{array} \Rightarrow D = \begin{array}{c|ccc} & C & A & B \\ \hline C & 0 & 3 & 1 \\ A & 2 & 0 & 1 \\ B & 1 & 1 & 0 \end{array}$$

Figure 15: Possible Way of Swapping Facilities

Shipping cost= 0*0 + 3*1 + 1*1 + 2*2 + 0*0 + 2*1 + 1*3 + 1*1 + 0*0 = **14**.

3.2 Multi-objective Quadratic Assignment Problem (mQAP)

Knowles and Corn (Knowles, J.D. and Corne, D.W, 2002), have proposed a variation of QAP so called Multi-objective Quadratic Assignment Problem (mQAP). The mQAP has multiple flow matrices and still only one distance matrix. In mQAP all objectives are a simple standard QAP individually and each one uses its own flow matrix. Hence, placing well-located facilities with respect to one of the flow matrices (considering one objective) may lead facilities to have a poor placement in relation to other objective(s).

Given a distance matrix D=$(d_{ij})_{n*n}$ for n locations, and m flow matrices $F^k = \{f_{ij}^k\}_{n*n*m}$, where, k=1, …, m and the mQAP is to minimize the following objective functions simultaneously:

$$min = C(\varphi) = \{C^1(\varphi), C^2(\varphi), \dots, C^m(\varphi)\}, \varphi \epsilon P_n \qquad (15)$$

$$C^k(\varphi) = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} f_{\varphi(\mathrm{i}),\varphi(\mathrm{j})}^k \quad 1 \leq k \leq m \qquad (16)$$

Where, n is the number of facilities, m is the number of objectives (flows), φ is a permutation of members from 1, …, n, P_n is the set of all permutations, $C(\varphi)$ is a vector of m objective function $C^k(\varphi)$, d_{ij} is the distance between locations i and j, and $f_{\varphi(\mathrm{i}),\varphi(\mathrm{j})}^k$ is thek^{th} flow between facilities $\varphi(i)$ and $\varphi(j)$.

In the case of conflicting objectives, there is no solution φ^* which is optimal for all objective functions $C^k(\varphi)$, φ =1, …, m. Instead, the optimal solution φ^* to the mQAP in (Eq. 14) is often designed as like the trade-off solution in terms of Pareto Optimality which is described in section 1.1.2.

Chapter 4

PROPOSE ARTIFICIAL BEE COLONY FOR QAP

4.1 Introduction

In the context of ABC Algorithm we succeed to propose a new ABC. In new modified ABC Algorithm, both Employed and Onlooker phases are the same as the standard ABC Algorithm. With the difference that a limited archive added (Adnan Acan, Ahmet Ünveren, 2005) to keep the fittest last solutions. Moreover this algorithm eliminates Scout phase and instead adds a new phase namely Crossover phase. The new phase is located after the Onlooker phase. In each iteration the best solutions send to archive and then these solutions sort in descending order. It means that the best solution so far has the highest order in archive and so on. Then crossover phase starts from the top solutions in archive and combine solutions pairwise to bottom respectively (Figure 16). Accordingly, the two solutions (parents) with the closest fitness value will create the new individual (offspring).

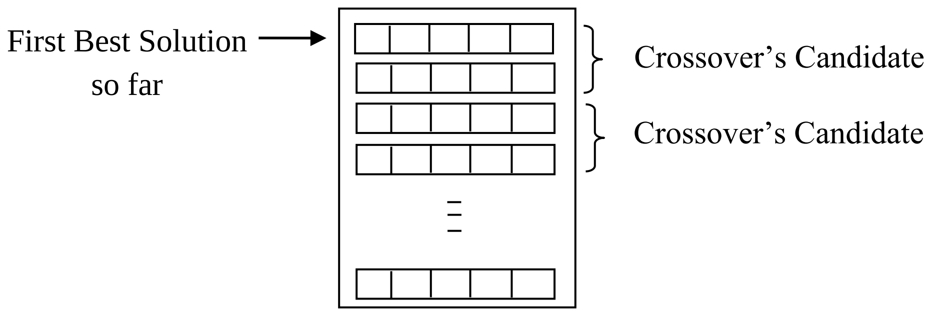


Figure16: Archive which Contains Fittest Solutions so far

The crossover operator has different forms in the number of parents that it selects to combine and the number of offspring that it creates after combination. Furthermore, this operator is used different in different form of solutions. For example, the crossover that done in binary form of solutions can not directly use in permutation form of solutions. It should consider together with some constraints to create the legal form of solutions, otherwise the created solutions will be a contrary form of solution.

In one of the standard form of crossover operator (Figure 17), for making the offspring, firstly, two solutions (parents) are selected then, two different random points are created and according to these points the middle of parents are changed with each other but, the both sides of parents do not touch and shift to offspring unchanged. Here, after each combination two offspring are created.

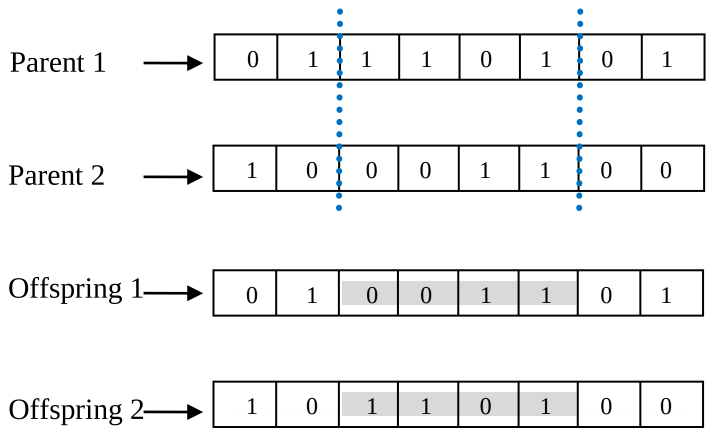


Figure 17: Standard Form of Crossover Done on Two Parents

The crossover operator for combining two permutation form of solutions is different with the one for real or binary solutions. In QAP, the solution is a permutation form or in another words it is an order of assigning the numbers (facilities' number) to some locations. In this form, each solution contains some digits (the number of digits depends on the dimension of the problem) and each digit repeats once. The point is, the solution(s) created from the current solution(s) should be a new acceptable solution in regard to all constraints. To this point, some criteria should be considered to meet the demand in each perturbation.

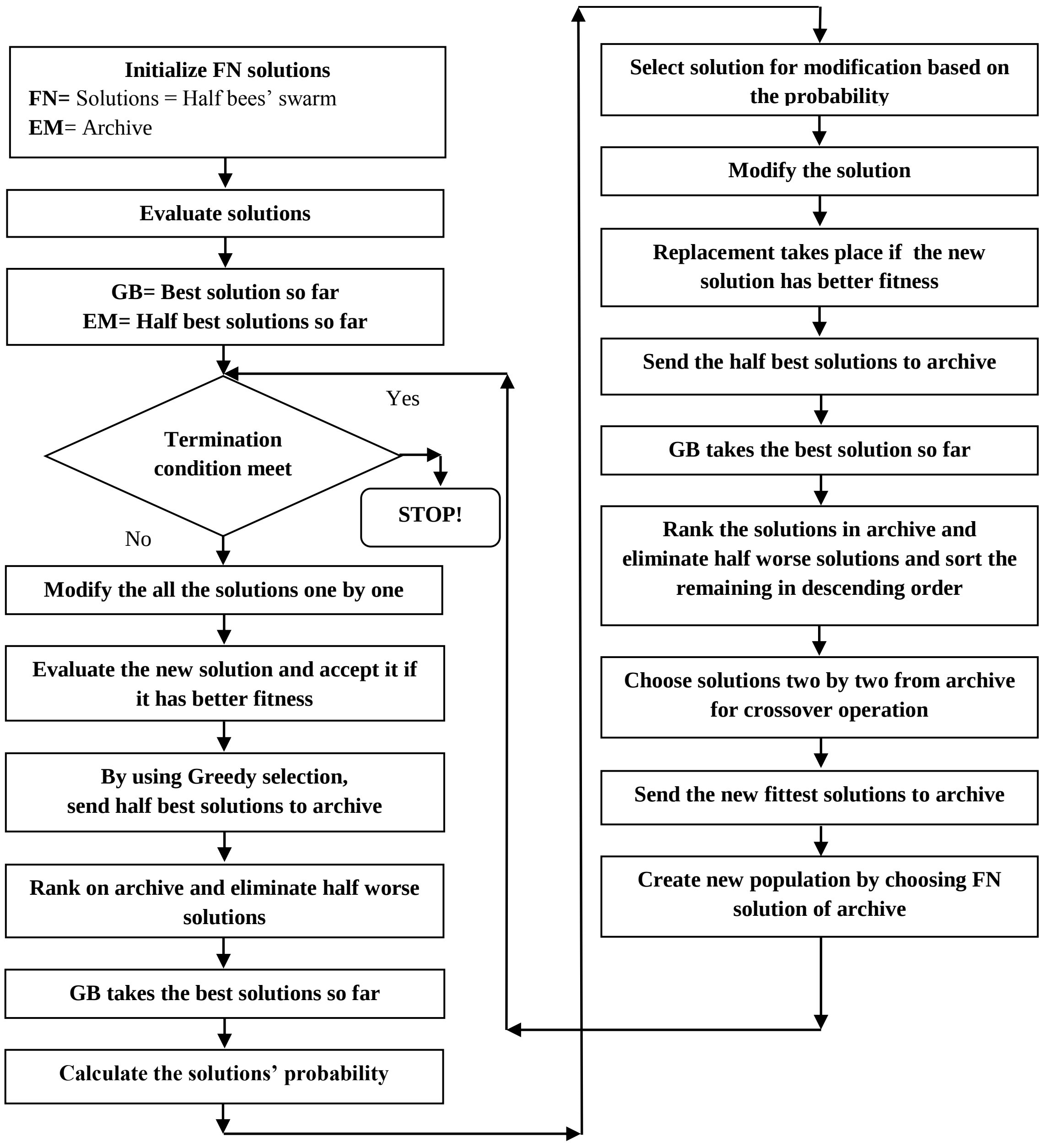
The new sophisticated crossover is utilized to avoid the perturbation. This mechanism does on the solutions in the archive and then the suitable combination of solutions is selected. These solutions are compared with the previous solutions and then replacement takes place if better fitness is achieved. The required constraints for getting the legal solutions after each perturbation by the new crossover are considered as well.

4.2 Proposed Artificial Bee Colony with its Algorithm and Flowchart

The modified ABC Algorithm has an archive memory to maintain the fittest solutions so far. In each cycle, the subsequent population is elected from a linkage of current population together with the solutions in the archive. Hence, in this case, each cycle starts with partially better solutions than the former cycle. Size of archive is considered as the same as the size of population. Furthermore, in modified ABC a new crossover is proposed. It considered as for improving the exploitation in ABC. The crossover is used to perform on the solutions in archive. The power of this exploitation is not expected in many crossovers which the operation has been done completely random. The proposed algorithm is used as a solution of Quadratic Assignment Problem (QAP) and its pseudo code and related flowchart are given in the next two pages respectively.

Suppose the total number of bees is *SN*, the number of iterations is Max-cycle, EM keeps the *SN* number of best solutions in each iteration, and *GB* keeps the global best solution so far. Considering these definitions, the pseudo code and the flowchart for the proposed Artificial Bee Colony for the QAP is given below:

```
(1)  Initialization:
        (1.1) Find the SN/2 random number ( $X_i = \{x_1, x_2, \dots, x_m\}$ ) of feasible solution
              as an initial population.
        (1.2)  Evaluate the population.
        (1.3)  Select the best solution and keep it in GB.
        (1.4)  send half best solutions to EM.
Cycle=1;
While (Cycle<Max-cycle)
(2)  Employed bee phase:
        For i=1 to SN/2,
             (2.1)  Do modification over all solutions $X_i$, for i=1 to SN/2.
             (2.2)  Evaluate the population.
             (2.3)  If the fitness of solution is better than previous one then replace it,
                    otherwise $X_i$=previous solution.
        End;
        (2.4)  Send half best solutions to EM.
        (2.5)  Do rank mechanism on EM and save half solutions of archive.
        (2.6)  remove half worst solutions.
        (2.7)  Evaluate the probability of solutions' goodness $P_i = \frac{f_{i+1}}{\sum_{n=1}^{N} f_{i+n}}$.
        (2.8)  Keep the best solution so far in GB.
(3): Onlooker bee phase:
        (3.1)  Do SN/2 number modification on population (selection is based on $P_i$).
        (3.2)  Evaluate the fitness.
        (3.3)  If the fitness of new solution is better than current then replace it with the
               current, otherwise keep the current solution.
(4): Crossover phase:
        (4.1)  Do crossover on all solutions in EM.
        (4.2)  Evaluate the fitness of new solutions and keep the promising ones.
        (4.3)  Compare the best solution so far with GB and keep the best in GB.
Cycle=Cycle+1
End while;
```



Initialize FN solutions
FN= Solutions = Half bees' swarm
EM= Archive
Evaluate solutions
GB= Best solution so far
EM= Half best solutions so far
Termination condition meet
Yes
STOP!
No
Modify the all the solutions one by one
Evaluate the new solution and accept it if it has better fitness
By using Greedy selection, send half best solutions to archive
Rank on archive and eliminate half worse solutions
GB takes the best solutions so far
Calculate the solutions' probability
Select solution for modification based on the probability
Modify the solution
Replacement takes place if the new solution has better fitness
Send the half best solutions to archive
GB takes the best solution so far
Rank the solutions in archive and eliminate half worse solutions and sort the remaining in descending order
Choose solutions two by two from archive for crossover operation
Send the new fittest solutions to archive
Create new population by choosing FN solution of archive

Figure 18: Proposed Artificial Bee Colony Algorithm for QAP

4.2.1 Initialization

In initialization phase each solution (bee) considered as a new randomly initial solution $X_i = \{x_1, x_2, \dots, x_m\}$ that is a sequential allocation of facilities to locations (a permutation form). Then the fitness is evaluated by multiplying the F_{n*n} (flow) matrix with the D_{n*n} (distance) matrix based on each permutation. The initialization phase is continued for *FN* times. *FN* is equal to the half of SN (the bees' population). The best solution is taken from this step and *GB* keeps it. Also the Extent Memory (EA) keeps the half best solutions in this phase.

4.2.2 Employed Phase

Employed phase changes the solutions with shift mutation (section 5.2.1). Then a comparison between new solution and previous ones takes place in which, whenever a better solution attains the replacement occurs. This selection is done with Greedy Selection (GS) (Goran Dimic, Nicholas D. Sidiropoulos, 2005) mechanism. Also the best solution in $V_i = \{x_{i1}, x_{i2}, \dots, x_{in}\}$ is compared with *GB* and if it finds a better solution, the *GB* will take it. In minimization problems, the solution with lower fitness value is counted as better solution and in maximization problems the solution with higher fitness value. Furthermore, the EA contains the half best solutions from the Initialization phase and takes half best solutions from the Employed phase. Then all the solutions move to the next phase with a considered related probability based on their fitness among all solution in the population. Clearly the highest fitness in the population gets higher probability of selection.

4.2. 3 Onlooker Phase

In this phase the selection mechanism is based on the probability achieved in Employed phase and it is accomplished with the Roulette Wheel Selection (RWS) (R.Sivaraj et al., 2011) Method. Then the Neighbor Change (section 5.2.3) mutation, changes the solutions and immediately the replacement takes place if the new solution becomes better than the old one. Considering that the size of EA is earmarked same as the size of *SN* and in this phase EA is full with keeping half best solution from Employed and half from Onlooker phase. Due to restricted capacity of EA the Rank Mechanism is used for selecting half best solutions. Finally, the best solution is compared with *GB* and EA keeps half of the best solutions in this phase.

4.2.4 Crossover Phase

Considering the mentioned information about the crossover operator, the new crossover is designed to select two parents and create one offspring. Here, the individual' elements which are the same value and position in both parents are transferred to offspring with keeping the element and its position in the solution. Finally, if still there is/are some element(s) in offspring (it means that there were some elements in parents which had not the same value and position) it/they create randomly with respect to all constraints (Figure 19).

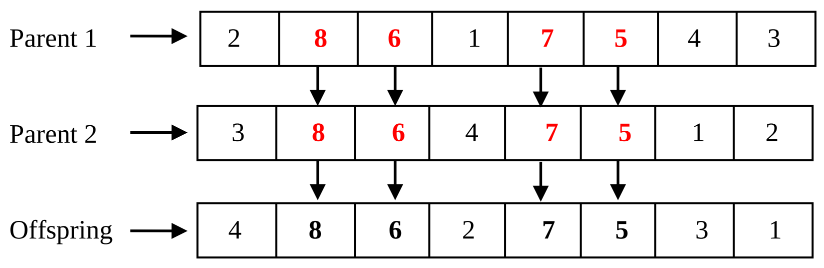


Figure 19: Proposed Crossover Operator

With this approach, in the last cycles, it is expected to have the solutions with partially rightful sequence of allocation or in another word having offspring with more proper elements.

After creating each new solution (offspring), the next step is to measure it and if its fitness is better than the parents' one, then, it should send to archive. At the end of each cycle, if the archive has more solutions than its predefined size (the size is considered same as the population size), then the next population will choose from the archive by ranking the solutions and taking the population size of solutions. Since, the new fittest solutions build the new population it is expectable to have convergence to the optimum solution step by step.

Since obtaining the appropriate solutions is not often expected from the scout phase and experience has shown it has not enough effect, so, this study eliminates this phase. Instead the shift mutation is used to avoid sticking at the local optima and having diversity in search space.

Note that, the experimental results of proposed ABC for the solution of QAP are analyzed in chapter 6.

Chapter 5

ARTIFICIAL MULTI-OBJECTIVE BEE COLONY (MOABC) ALGORITHM FOR mQAP

5.1 Introduction

The mutation operator (R. Storn, K. Price, 2010) is a genetic operator that alters the solution in one or more elements compare with the initial state. This operator is able to change the solution good enough and get the better result compare with the previous solution. Hence, the Multi-objective Artificial Bee Colony (MOABC) Algorithm can arrive at a better situation by using the mutation operator. In MOABC we have used a variety of mutation during the evolution search by defined mutation rate. This rate is considered low; otherwise, the search will turn to a random search.

Tabu Search (TS) Algorithm (Glover F, 1986) also is a powerful neighbor search method. The main idea behind the TS is to avoid of visiting some solutions in each iteration. It has been done with considering a tabu list in which the solutions with worse history are kept and the algorithm does not allow to visit these solutions until some criteria change their position as eligible solutions and then, they leave the tabu list immediately. The new version of TS Algorithm is Robust Tabu Search (RTS) Algorithm (Taillard, E., 1991) and it also is used in MOABC Algorithm. In RTS Algorithm there are some modifications to enable the RTS to be useful for multi-objective Quadratic Assignment Problem (mQAP).

With considering the efficiency and complexity in MOABC Algorithm, by combining the RTS Algorithm and using the different kind of mutations, we have tried to propose a powerful algorithm with high efficiency and low complexity as much as possible.

The Crowding Distance, Fitness function, Employed and Unemployed Bees are used in MOABC Algorithm and they are explained in chapter two. However, the mutation kinds and the Tabu Search are explained in this chapter in more detail.

5.2 Mutation Operator

The MOABC Algorithm, after the initialization phase with considering the three phases (Employed, Onlooker and Crossover phases) repeats these kinds of operations continuously to create new individuals from the current ones. Here, a well-known operator so called mutation is used to alter some gene(s) in an individual (parent) to create a new individual (offspring). Figure 20 shows a general form of mutation operator which can be used for QAPs' solution.

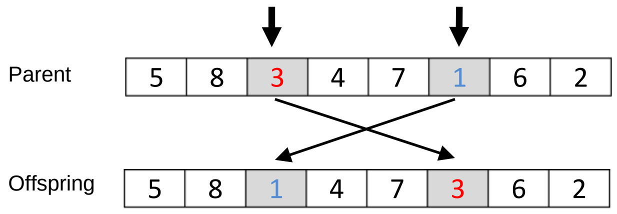


Figure 20: General Form of Mutation

Figure 20 shows that, with perturbing one parent, one offspring is created. The idea is to swap two elements with each other (elements 3 and 8 in this Figure 20).

There are varieties of mutations that this study has used some, such as: Shift Mutation, Swap Mutation, Neighbour Interchange Mutation and Mixed Mutation. These kinds of mutation are explained in detail below.

5.2.1 Shift Mutation

Commonly, in shift mutation two random integer numbers are created. The range of these two numbers is between one and dimension amount. One of these number save in a temporary memory. Then all elements between these two number shift forward or backward. The direction of shifting depends on the number maintained in the temporary memory. If the bigger digit saved in archive, the direction is forward and vice versa. Lastly, the element of the saved number in the temporary memory will assigned instead of the last shifted element. Figure 21 shows this process.

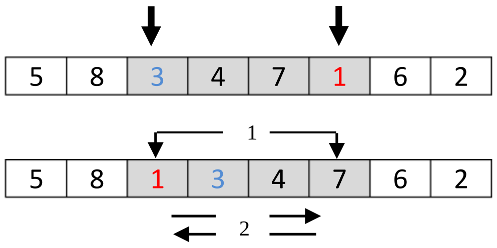


Figure 21: Shift Mutation

5.2.2 Swap Mutation

In this case two random integer numbers between one and dimension amount are created and then the elements of these two positions in solutions swaps (change with each other) immediately. Figure 22 shows the operation.

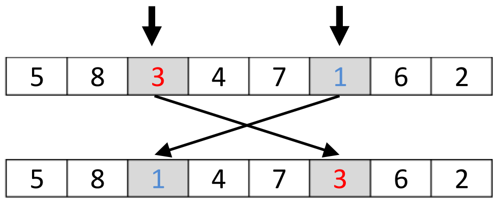


Figure 22: Swap Mutation

5.2.3 Neighbour Interchange Mutation

Here, just one random integer number between one and dimension amount is created and the element of this number will swap with the next element in forward cases or with the previous one in backward case immediately. The direction of swapping is determined by created random number in the range of [0, 1]. If the number is less than 0.5 the direction is in forward case, otherwise, in backward case. Figure 23 shows this operation:

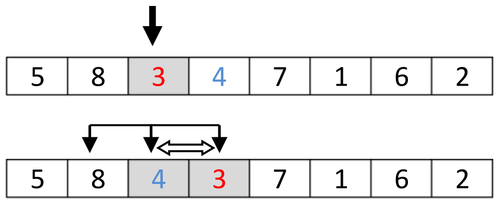


Figure 23: Neighbour Interchange Mutation

5.2.4 Mixed Mutation

This case attempts to have a mixture of mutations in algorithm. To this point in each cycle different kind of mutation (mutations which are considered above) takes place. Selection of mutation is random completely. It means that each cycle modify solutions with respect to different random selected mutation.

5.3 Robust Tabu Search (RTS)

Tabu Search (TS) is one of the earliest algorithms for optimization problems. TS algorithm was proposed by Glover (Glover F, 1986) and then formalized (Fred Glover, 1989) and (Fred Glover, 1990) and further enhanced with more detail by Glover & Laguna (Glover & Laguna, 1997), as a local search method in optimization. Suppose *S* is a collection of solutions (s_1, s_2, ...) of a hard problem to solve with objective (fitness) function $f: S \rightarrow R^1$; as well as, suppose $N: S \rightarrow 2^s$ be a vicinity subordinate which defines for each $s \in S$ a set $N(s) \subseteq S$ - a set of neighboring solutions of S. Each solution $s' \in N(s)$ can be attained from S by move operation (Alfonsas M., 2005). Neighbor solutions are constructed to exploit if there is a better adjacent solution that can be reached from the current solution. If so, the current solution will swap with this solution. Among the period of extracting the solutions a tabu list is to record a subset of the moves in a neighborhood as forbidden. The size of the tabu list is an important issue in TS. If it is considered too small then a cycling will occur, whereas if it is too large, it will restrict the search to promising regions (Alfonsas M. 2005). Therefore it should update after a while. TS has an important exception named aspiration criteria, which contains some conditions that permit the approach to override the run if required. These criteria will be met when a tabu move has a favorable situation or sufficient attraction.

Robust Tabu Search (RTS) (Taillard, E, 1991) is a variation of standard TS and here with considering more constraints we have used it for Quadratic Assignment Problem (QAP). The RTS has two main differences compare with the standard form of TS. Firstly, RTS has random variables in tenure the tabu elements instead of a static value. Secondly, the RTS Algorithm features a long-term diversification method which favors moves that have not been performed in a long time. If a move would assign to facilities to locations that both have not been assigned to for a long number of iterations, then the move is performed regardless of quality. If there are multiple of these moves in one iteration the best one is performed. The proposed algorithm for the solution of mQAP will be given in the following sections.

Note that, theoretically, the number of PO solutions can be infinite. Since, the ultimate purpose of a Multi-objective Optimization (MO) Algorithm is to provide a set of solutions to choose by user from, it is necessary to limit the size of this set to be usable by the user (S. Bandyopadhyay, et al, 2008). In the last part of MOABC Algorithm it is checked, if the number of obtained solutions is more than the predefined number, by using the Crowding Distance (CD) (Carlo R. et al, 2005) some solutions are eliminated to maintain the diversity and have a variety of solutions instead of focusing in some parts of search space. To calculate the CD there are some techniques, one shown in Figure 24 in which, solutions are shown with points. For computation, ordering the non-dominated solutions in archive in ascending order with respect to each objective function is required.

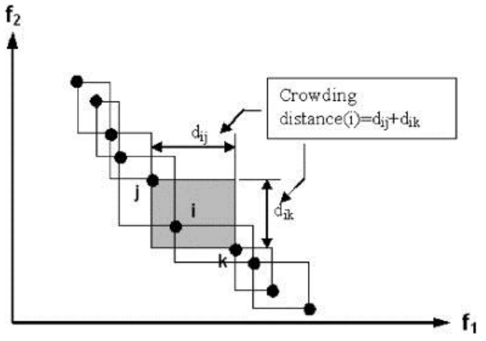


Figure 24: Crowding Distance

5.4 MOABC Algorithm with Different Variation

In this study, different updating mechanisms are applied in MOABC Algorithm to improve the performance of this algorithm for the solution of mQAP. To this point, we have used different sorts of mutation operators such as: shift, swap, and neighbor interchange mutations. On the other hand, the Robust Tabu Search (RTS), which is the powerful local search, was used to improve further the solutions. Consequently, it is realized that, adding the shift mutation and RTS to MOABC Algorithm, can be the desirable modification to get closer to the goal.

Figure 25 shows a flowchart of MOABC Algorithm. Here, in the initialization phase, a population of N number of solutions is created randomly and S_0 keeps these solutions. Then, these solutions are evaluated and non-dominated solutions are kept in an External Archive (EA). From this point, the population is modified in employed, onlooker, and RTS phases to get better solutions. Solutions created by employed and onlooker phase are sent to the set C_t. Also, solutions created by RTS are send to the set S_t, if better solutions are gained. After each phase, new solutions are evaluated and if they dominate the current solutions in the mentioned sets and the current population, the replacements takes place and non-dominated solutions are sent to EA and dominated solutions are removed (update EA). Finally, new generation (S_{t+1}) is extracted from the combination of S_t and C_t and the fittest solutions are taken. The last step uses crowding distance to remove the crowded solutions from the set of non-dominated solutions and maintaining a uniform set of solutions with respect to the diversity.

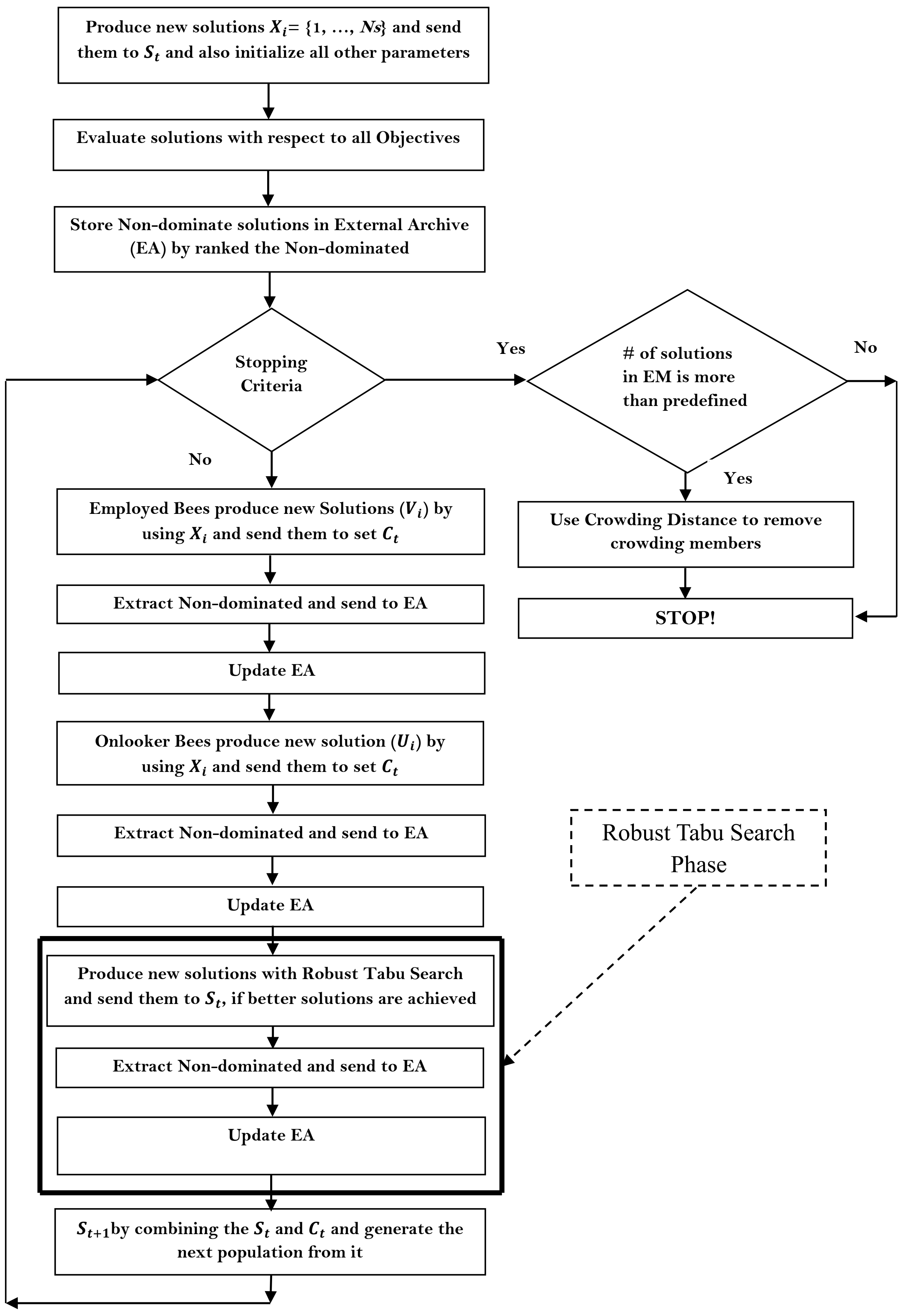

Produce new solutions X_i= {1, …, Ns} and send them to S_t and also initialize all other parameters
Evaluate solutions with respect to all Objectives
Store Non-dominate solutions in External Archive (EA) by ranked the Non-dominated
Stopping Criteria
Yes
No
# of solutions in EM is more than predefined
No
Yes
Use Crowding Distance to remove crowding members
STOP!
Employed Bees produce new Solutions (V_i) by using X_i and send them to set C_t
Extract Non-dominated and send to EA
Update EA
Onlooker Bees produce new solution (U_i) by using X_i and send them to set C_t
Extract Non-dominated and send to EA
Update EA
Robust Tabu Search Phase
Produce new solutions with Robust Tabu Search and send them to S_t, if better solutions are achieved
Extract Non-dominated and send to EA
Update EA
$S_{t+1}$by combining the S_t and C_t and generate the next population from it

Figure 25: MOABC Algorithm Combined with RTS for mQAP

Chapter 6

EXPERIMENTAL RESULTS

6.1. Results for Proposed Artificial Bee Colony (ABC) Algorithm

In this study, we have improved the exploitation ability of the Single-objective Artificial Bee Colony (ABC). Then, we solved the Quadratic Assignment problem (QAP) problems available at QAPLIB as our benchmarks (http://www.seas.upenn.edu/qaplib/). Also, for Multi-objective Quadratic Assignment Problem we considered all the instances in http://www.cs.man.ac.uk/~jknowles/mQAP/. Finally, the results have been compared with the results in mentioned sites and some other modified MOABC Algorithms.

The results of single-objective ABC have compared with MABC and Crossover ABC (CABC) (proposed algorithm). In the MABC, by using an External Memory (EM), in each cycle, only best results are gathered and without any perturbation are sent to the next iteration and are used as starting points. Eventually, we have established that the results of CABC are much better than the other two mentioned algorithms. The CABC successfully gained the exact solutions in benchmarks like bur26e, bur26h in a little time and small cycle, which is not expected of ABC. Also, for high dimensions the difference between optimal solution and CABC is much less than other algorithms. The results of ABC, MABC and CABC Algorithms are shown in table1. Equally, the Optimum results, Standard Deviations (SD) obtained results by CABC, Best and Worst results for some benchmarks are found in Table 2.

Table 1: Results of ABC, MABC and CABC Algorithms

Benchmark	Optimum	ABC	MABC	CABC
tai12a	224416	240972	224416	224416
tai15a	388214	417718	388214	388214
tai20a	703482	799284	720688	707178
tai25a	1167256	1319594	1207270	1187682
tai30a	1818146	2052574	1872120	1839112
tai35a	2422002	2759766	2514582	2469028
tai40a	3139370	3576762	3266370	3200988
tai50a	4938796	5171290	5144154	5055748
tai60a	7205962	7514246	7552688	7316578
tai80a	13499184	15200966	14160626	13796940
tai100a	21052466	23506604	22091632	21405442
tai150b	498896643	619614546	525594052	510964347
chr12a	9552	12156	9552	9552
chr15c	9504	18580	9780	9504
chr18a	11098	27120	13450	11098
chr20b	2298	4730	2688	2368
chr25a	3796	11012	4924	3946
bur26a	5426670	5552273	5427776	5426670
bur26e	5386879	5535955	5392187	5386879
bur26g	10117172	10366184	10118819	10117172
bur26h	7098658	7276495	7098905	7098658

Table 2: Standard Deviations, Best and Worst Results for Instances

Benchmark	Optimum	Best	Worst	SD
tai12a	224416	224416	224416	0
tai15a	388214	388214	388214	0
tai20a	703482	707178	714218	2693
tai25a	1167256	1173672	1204380	9111.58
tai30a	1818146	1843434	1875768	8636.73
tai35a	2422002	2466054	2520282	14831.48
tai40a	3139370	3214310	3265418	13176.82
tai50a	4938796	5059410	5144144	25788.74
tai60a	7205962	7416902	7482498	21092.23
tai80a	13499184	13975506	14105352	35592.49
tai100a	21052466	21405442	21620214	72655
tai150b	498896643	510655361	524042016	3554423.80
chr12a	9552	9552	9552	0
chr15c	9504	9504	9504	0
chr18a	11098	11098	12432	593.19
chr20b	2298	2412	2536	44.38
chr25a	3796	3946	4922	299
bur26a	5426670	5426670	5432449	2358.02
bur26e	5386879	5386879	5386879	0
bur26g	10117172	10117172	10118141	474.71
bur26h	7098658	7098658	7098658	0

6.2 Results for Proposed Multi-objective ABC Algorithm

To illustrate the performance of the MOABC combined with Robust Tabu Search (RTS), we have been considering the instances available in the Knowles' dataset in http://www.cs.man.ac.uk/~jknowles/mQAP/#suites. Then, we prepared four different sorts of MOABC Algorithms with different updating methods. Finally, the results of modified MOABC Algorithm are compared with the results of these four algorithms. It is noticeable that, in the mentioned site, only 10-dimensional problems' Optimum Pareto fronts are provided and for the rest of problems (twenty, thirty, fifty, and seventy-five dimensions) we have tried to provide them. For this purpose, all the obtained results from the four mentioned algorithms above together with the results of modified MOABC Algorithm, are gathered and fed them to the modified MOABC Algorithm and run this algorithm for a two times cycle to get the best results in Pareto-optimum front as much as possible. In the figure s, the number of non-dominated solutions obtained with each algorithm is written in the Y-axis. Also, Title of the instances, their associated Pareto front, types, number of locations and facilities are provided in Table 4. In the Table 5, there are results of some metrics which are done on the obtained Pareto front by modified MOABC. These metrics are: Error Ration (ER), Convergence Metric (CM), Generational Distance (GD), Hyper Volume (I_HypVol), and Divergence Metric (DM).

Table 3: Positions and Abbreviations of Modified MOABC Algorithms in Figures

MOABC Algorithms	Abbreviation in Figure	Pareto fronts' Position in Figures
Optima Pareto Front	Optimum	Upper left
Neighbour interchange	Ni-mutation	Upper middle
Swap Mutation	Sw-mutation	Upper right
Robust Tabu Search	RoTS	Bottom Left
Mixed Mutation	Mi-mutation	Bottom middle
Shift Mutation	Sh-mutation	Bottom right

Table 4: Benchmarks and Their Associated Pareto Front in Figures

Pareto Front	Instance Name	Instances Type	Location No.	Objective No.
Figure 26	KC10-2fl-1uni	Uniform Instances	10	2
Figure 27	KC10-2fl-2uni	Uniform Instances	10	2
Figure 28	KC10-2fl-3uni	Uniform Instances	10	2
Figure 29	KC20-2fl-1uni	Uniform Instances	20	2
Figure 30	KC20-2fl-2uni	Uniform Instances	20	2
Figure 31	KC20-2fl-3uni	Uniform Instances	20	2
Figure 32	KC30-3fl-1uni	Uniform Instances	30	3
Figure 33	KC30-3fl-2uni	Uniform Instances	30	3
Figure 34	KC30-3fl-3uni	Uniform Instances	30	3
Figure 35	KC50_2fl_1uni	Uniform Instances	50	2
Figure 36	KC50_2fl_2uni	Uniform Instances	50	2
Figure 37	KC50_2fl_3uni	Uniform Instances	50	2
Figure 38	KC75_3fl_1uni	Uniform Instances	75	3
Figure 39	KC75_3fl_2uni	Uniform Instances	75	3
Figure 40	KC10-2fl-1rl	Real Instances	10	2
Figure 41	KC10-2fl-2rl	Real Instances	10	2
Figure 42	KC10-2fl-3rl	Real Instances	10	3
Figure 43	KC10-2fl-4rl	Real Instances	10	2
Figure 44	KC10-2fl-5rl	Real Instances	10	2
Figure 45	KC20-2fl-1rl	Real Instances	20	2
Figure 46	KC20-2fl-2rl	Real Instances	20	2
Figure 47	KC20-2fl-3rl	Real Instances	20	2
Figure 48	KC20-2fl-4rl	Real Instances	20	2
Figure 49	KC20-2fl-5rl	Real Instances	20	2
Figure 50	KC30-3fl-1rl	Real Instances	30	3
Figure 51	KC30-3fl-2rl	Real Instances	30	3
Figure 52	KC30-3fl-3rl	Real Instances	30	3
Figure 53	KC50_2fl_1rl	Real Instances	50	2
Figure 54	KC50_2fl_2rl	Real Instances	50	2
Figure 55	KC50_2fl_3rl	Real Instances	50	2

Table 5: Comparison of Obtained Results by Modified MOABC through Metrics

Instances Name	ER	CM	GD	I_HypVol	DM
KC10-2fl-1uni	0	0	0	0.0293	1
KC10-2fl-2uni	0	0	0	0	1
KC10-2fl-3uni	0.8923	5.3708e+003	613.6236	0.0663	1
KC20-2fl-1uni	0.1452	165.5229	70.7021	0.1313	0.3913
KC20-2fl-2uni	0.6000	1.1480e+003	921.7815	0.3737	0.4286
KC20-2fl-3uni	0.0332	35.9622	15.3435	15.3435	0.3185
KC30-3fl-1uni	0.7121	6.9891e+003	1.1496e+003	0.0173	0.3210
KC30-3fl-2uni	0.9286	1.5259e+004	3.8563e+003	0.1241	0.4474
KC30-3fl-3uni	1	8.7634e+003	939.0832	0.0095	0.3359
KC50_2fl_1uni	0.9239	1.6305e+004	2.4274e+003	0.2891	0.2376
KC50_2fl_2uni	0.6667	1.0874e+004	7.5535e+003	0.0622	0.3077
KC50_2fl_3uni	0.7656	4.6964e+003	301.6019	0.0507	0.2003
KC75_3fl_1uni	0.6759	2.0409e+004	2.7342e+003	0.0365	0.2273
KC75_3fl_2uni	0.9000	4.1172e+004	1.2035e+004	0.2708	0.2273
KC10-2fl-1rl	0	0	0	0.3381	1
KC10-2fl-2rl	0	0	0	0.1117	1
KC10-2fl-3rl	0	0	0	0.3550	1
KC10-2fl-4rl	0	0	0	0.2500	1
KC10-2fl-5rl	0	0	0	0.6414	1
KC20-2fl-1rl	0.0215	1.6926e+003	1.4351e+003	0.0369	0.2553
KC20-2fl-2rl	0.2149	1.1570e+004	3.0547e+003	0.0451	0.1060
KC20-2fl-3rl	0.0135	3.1485e+003	1.9651e+003	0.0761	0.1398
KC20-2fl-4rl	0.2193	3.1149e+005	1.3672e+005	0.0502	0.0970
KC20-2fl-5rl	0.2885	1.0735e+005	2.1251e+004	0	0.1758
KC30-3fl-1rl	0.9080	1.9441e+005	1.8779e+004	0.0467	0.0690
KC30-3fl-2rl	0.9457	1.0368e+005	1.0758e+004	0.1386	0.0577
KC30-3fl-3rl	0.5068	8.0210e+004	1.0260e+004	0.0581	0.0461
KC50_2fl_1rl	0.7838	2.8054e+005	1.9293e+004	0.1404	0.0892
KC50_2fl_2rl	0.5475	1.6912e+005	1.6035e+004	0.0167	0.0863
KC50_2fl_3rl	0.6553	1.5697e+005	9.5020e+003	0.0040	0.0571

Table 5 clearly shows the success of modified MOABC Algorithm in divergence and convergence aspects. It is shown the modified MOABC Algorithm has the exact results in some dimensional problem such as KC10-2fl-1uni, KC10-2fl-2uni, etc. by having the value zero in most metrics and one in DM. Furthermore, the results of other instances in higher dimensional problems are also admirable. In following, because the trends of instances are almost similar, only three figures are picked for more explanation below and the rest of figures are available in Appendix.

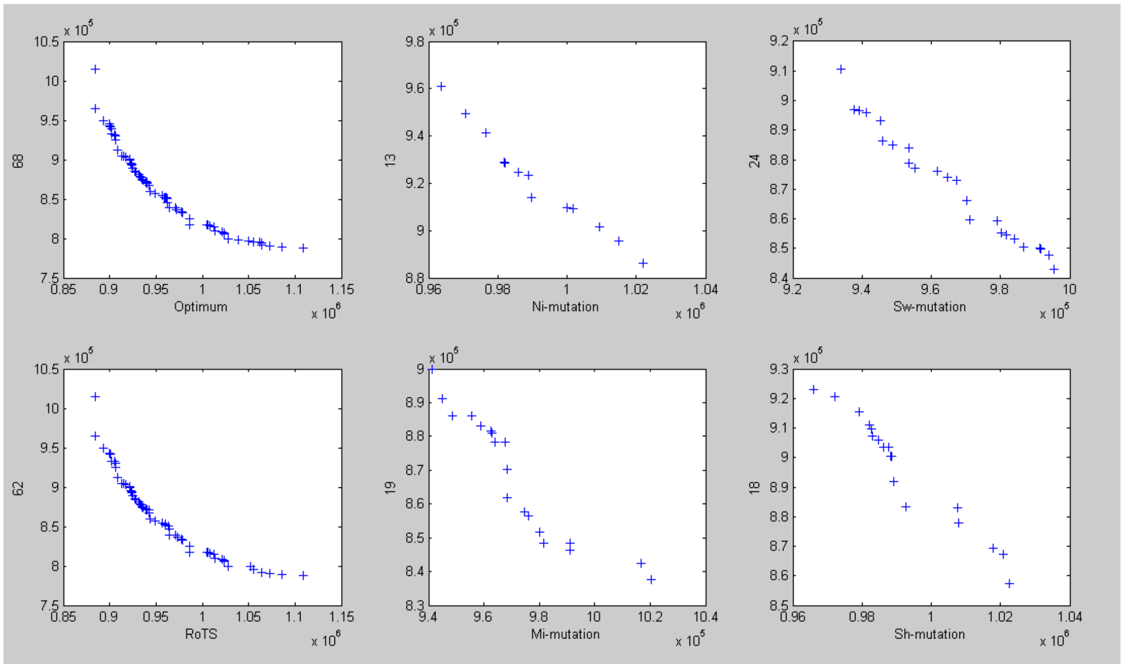


Figure 29: Pareto Front for KC20-2fl-1uni

In this figure, non-dominated solution sets provided by modified MOABC combined with: Neighbour Interchange Mutation (Ni-mutation), Swap Mutation (Sw-mutation), Robust Tabu Search (RoT), Mixed Mutation (Mi-mutation), and Shift Mutation (Sh-mutation) are applied to KC20-2fl-1uni uniform instance with 20 locations are shown. In this case, the non-dominated solutions found by RoTS is very close to the ones on the Optimum (Pareto front), however, the superiority in the success of RoTS is better seen by considering the number of non-dominated solutions. The number of solutions on the provided Pareto front is 68, 62 for RoTS, 13 for Ni-mutation, 24 for Sw-mutation, 19 for Mi-mutation, and 18 for Sh-mutation.

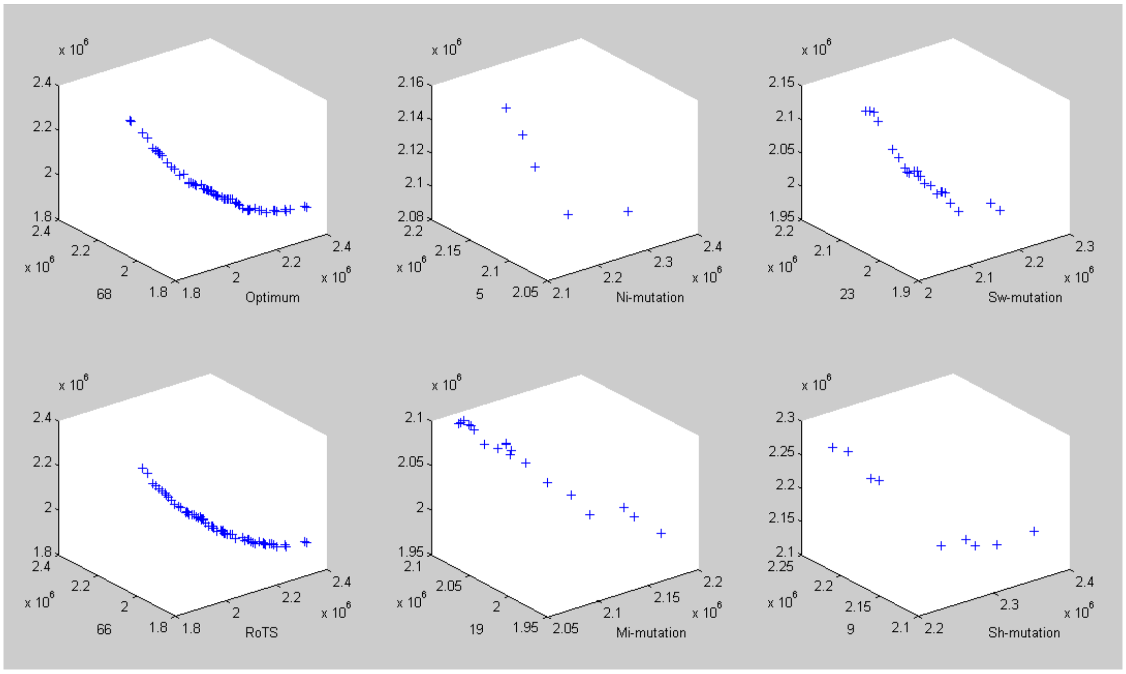


Figure 32: Pareto Front for KC30-3fl-1uni

Here, non-dominated solution sets provided by mentioned modified MOABC Algorithms are applied to KC30-3fl-1uni uniform instance with 30 locations are shown and 3 objectives (flow). Again it is clear that, non-dominated solutions found by RoTS are very close to the Optimum ones (Pareto front) and the accomplishment of the RoTS by considering the number of non-dominated solutions is more transparent. Where, the number of provided solutions by Pareto front is 68, 66 for RoTS, 5 for Ni-mutation, 23 for Sw-mutation, 19 for Mi-mutation, and 9 for Sh-mutation.

As it is illustrated in all figures, apart the MOABC combined with RoTS, all other modified MOABC do not have the similar performance to get closer to the Pareto-optimal front. Also, the number of non-dominated solution obtained by all other algorithms is not admirable compare with RoTS.

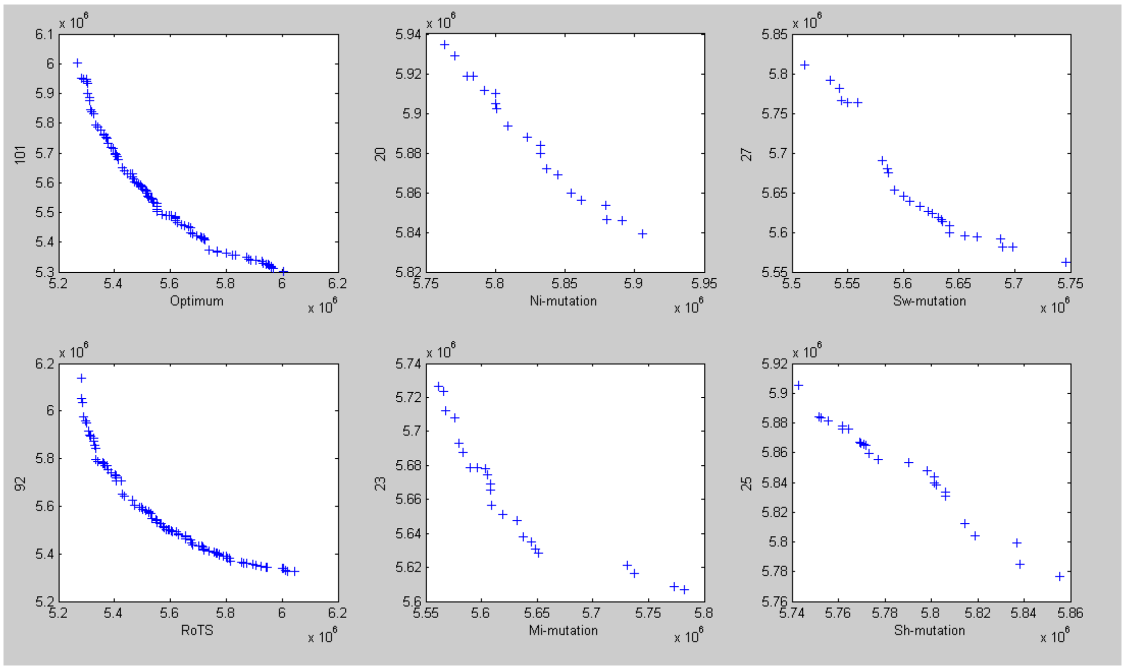


Figure 35: Pareto Front for KC50-2fl-1uni

Here, non-dominated solution sets provided by mentioned modified MOABC Algorithms are applied to KC50-2fl-1uni uniform instance with 50 locations are shown and 2 objectives (flow). Clearly, non-dominated solutions found by RoTS again are the most closest to the Optimum ones (Pareto front) and the power of the RoTS by considering the number of non-dominated solutions is more transparent. Where, the number of provided solutions by Pareto front is 101, 92 for RoTS, 20 for Ni-mutation, 27 for Sw-mutation, 23 for Mi-mutation, and 25 for Sh-mutation.

The rest of figures attained by all modified MOABC Algorithms are illustrated in the appendix. However, the performance in all algorithms is almost similar in all benchmarks.

Chapter 7

CONCLUSION

In this work, a modified Artificial Bee Colony (ABC) Algorithm for Quadratic Assignment Problem (QAP) and a modified Multi-objective ABC (MOABC) Algorithm for multi-objective QAP (mQAP) are presented. The proposed ABC Algorithm keeps the best solutions of each iteration for next iteration, instead of choosing a form at random. In this study we have eliminated the scout phase and used the shift mutation in order to avoid sticking in local optima. In this algorithm we have also added a new crossover operator to improve local search and exploitation ability of ABC Algorithm. These changes substantially improved the ABC Algorithm in balancing the exploitation and exploration power. By creating this modified ABC Algorithm we solved the QAP and can more confidently reach good solutions in high dimensional problems. Even reaching exact solutions in bur26e, bur26h dimensional problem and so on, which is unexpected for a heuristic algorithm, is achieved by this algorithm.

In the proposed MOABC Algorithm, we have used the variation of Tabu Search (TS) so called Robust Tabu Search (RoTS), together with the Shift Mutation operator. Doing this, we have managed to create a powerful algorithm in all aspects as a multi-objective problem solver. The mQAP has also been solved by using the MOABC Algorithm. The results show that we have arrived at an efficient algorithm in that it is a powerful tool in handling both convergence and divergence to the Pareto Optimum.

REFERENCES

Adnan Acan, Ahmet Ünveren. “Evolutionary multiobjective optimization with a segment based external memory support for the multiobjective quadratic assignment problem”. *Congress on Evolutionary Computation*. pp: 2723-2729. (2005).

Alexander Schrijver. "On the history of combinatorial optimization (till 1960), *Handbook of Discrete Optimization, Elsevier, Amsterdam*. pp: 1–68. (2005).

Alfonsas Misevivius. “A Tabu Search Algorithm for the Quadratic Assignment Problem”. *Springer Science, Computational Optimization and Applications,* 30, 95–111. pp: 96. (2005).

Ashlock, D. “Evolutionary Computation for Modeling and Optimization”. *Springer,* ISBN 0-387-22196-4. (2006).

Beni, G., Wang, J. Swarm. “Intelligence in Cellular Robotic Systems”, *Proceed. NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy,* pp: 26–30. (1989).

Carlos A. Coello Coello, Gary B. Lamont and David A. Van Veldhuizen. “Evolutionary Algorithms for Solving Multi-Objective Problems”. ISBN 978-0-387-33254-3. (2002).

Carlo R. Raquel and Prospero C. Naval, Jr. “An Effective Use of Crowding Distance in Multi-objective Particle Swarm Optimization”, *GECCO 05, Washington DC, USA.* (2005).

C. Lin and Y. Wang. “A new evolutionary algorithm for multi-objective optimization problems”. *ICIC*, vol.1, no.1. pp: 93-98. (2007).

Chunfan Xu, Haibin Duan, “Artificial bee colony (ABC) optimized edge potential function (EPF) approach to target recognition for low-altitude aircraft”, *Pattern Recognition Letters 31*, pp: 1759-1772. (2010).

Coello, C. A. C. “An updated survey of GA-based multi-objective optimization techniques”. *ACM Computing Surveys*, vol. 32, no. 2. pp: 109-143. (1999).

D. Karaboga. “An Idea Based on Honey Bee Swarm for Numerical Optimization”. *in Technical report-tr06, Erciyes University*. (2005).

Fred Glover. "Tabu Search - Part 1". *ORSA Journal on Computing 1 (2).* pp: 190-206. (1989).

Fred Glover. "Tabu Search - Part 2". *ORSA Journal on Computing 2 (1).* pp: 4-32. (1990).

Glover F. “Future paths for integer programming and links to artificial intelligence”. *Computers and Operations Research, 13*. pp: 533-549. (1986).

Glover, F., and Laguna, M. “Tabu Search”. *Kluwer Academic Publishers, Boston, MA.* (1997).

Goran Dimic Nicholas D. Sidiropoulos, “On Downlink Beamforming With Greedy User Selection: Performance Analysis and a Simple New Algorithm”, IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 53, NO. 10, (2005).

Greenwald, Bruce; Stiglitz, Joseph E. "Externalities in economies with imperfect information and incomplete markets". *Quarterly Journal of Economics* 101(2): 229-264. Doi: 10.2307/1891114. JSTOR 1891114. (1986).

K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. “A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA II”. *In Parallel Problem Solving from Nature VI* (PPSN VI). pp: 849-858. (2000).

K.E. Parsopoulos, M. N. Vrahatis. “Particle swarm optimization method in multiobjective problems”. *ACM New York, NY, USA*. pp: 603-607. (2002).

Kennedy, J. and Eberhart, R. C. “Particle swarm optimization”. *Proc. IEEE Int'l. Conf. on Neural Networks,* IV, 1942–1948. Piscataway, NJ: IEEE Service Center. (1995).

Kirkpatrick, S., Gelatt Jr., C., and Vecchi, M. “Optimization by simulated annealing”. *Science* 220, 4598, pp: 671-680. (1983).

Knowles, J.D. and Corne, D.W. “Towards landscape analyses to inform the design of a hybrid local search for the multiobjective quadratic assignment problem”. (2002).

Koopmans_Beckmann. “Assignment Problem and Location of Economic “. pp: 25. (1957).

Loiola, M. E., Abreu, N., Boaventura-netto, P., Hahn, P., Qurerido, T. “A survey for the quadratic assignment problem”. *European Journal of Operational Research,* 176. pp: 657-690. (2007).

Malcolm Yoke Hean Low and Chwee Seng Choo. “Application of Multi-Objective Bee Colony Optimization Algorithm to Automated Red Teaming” 978-1-4244-5771-7/09/$26.00 IEEE. (2009).

Maurice C. and James K. “The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space”. *Transaction on Evolutionary Computation*, IEEE. Vol. 6, No. 1, February. pp: 58-72. (2002).

MIT Press. "Evolutionary Computation". Vol. 1, No1. (1993).

Morse, P. M. and Feshbach, H. "Asymptotic Series; Method of Steepest Descent*". §4.6 in Methods of Theoretical Physics, Part I. New York: McGraw-Hill.* pp: 434-443. (1953).

Moscato, P. “On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms”. *Caltech Concurrent Computation Program,* C3P Report 826. (1989).

Omar Al Jadaan, Lakishmi Rajamani, C. R. Rao, “NON-DOMINATED RANKED GENETIC ALGORITHM FOR SOLVING MULTI-OBJECTIVE OPTIMIZATION PROBLEMS: NRGA”, *Journal of Theoretical and Applied Information Technology*, (2008).

R. Hedayatzadeh, B. Hasanizadeh, R. Akbari, K. Ziarati. “A Multi-Objective Artificial Bee Colony for Optimizing Multi-Objective Problems”. *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE),* Chengdu, China, (2010).

R. Sivaraj et al. “A REVIEW OF SELECTION METHODS IN GENETIC ALGORITHM” *International Journal of Engineering Science and Technology (IJEST),* ISSN : 0975-5462, Vol. 3. (2011).

R. Storn, K. Price. “Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces”, *J. Global Optim. pp:23.* (2010).

S. Bandyopadhyay, S. Saha, U. Maulik. “A Simulated Annealing-Based Multiobjective Optimization Algorithm: AMOSA”. Vol: 12, NO.3. pp: 270. (2008).

Sanaz Mostaghim and J¨urgen Teich. “Quad-trees: A Data Structure for Storing Pareto-sets in Multi-objective Evolutionary Algorithms with Elitism”. *Evolutionary Multi-objective Optimization. Springer*, ISBN 1-85233-787-7. pp: 81-104. (2005).

Taillard, E. “Robust taboo search for the quadratic assignment problem”. *Parallel Computing 17*. Pp: 443-455. (1991).

T.D. Seeley. “The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies”. ISBN-10: 0674953762. (1996).

Teodorovic, D., Orco, M.D. “Bee colony optimization – a cooperative learning approach to complex transportation problems”. *Advanced OR and AI Methods in Transportation.* (2005).

Zhang, X.-H., Meng, H.-Y., Jiao, L.-C. “Intelligent Particle Swarm Optimization in Multiobjective Optimization”. *Proc. IEEE Congress on Evolutionary Computation CEC*. Vol. 1. It cites paper C2. pp: 714-719. (2005).

APPENDIX

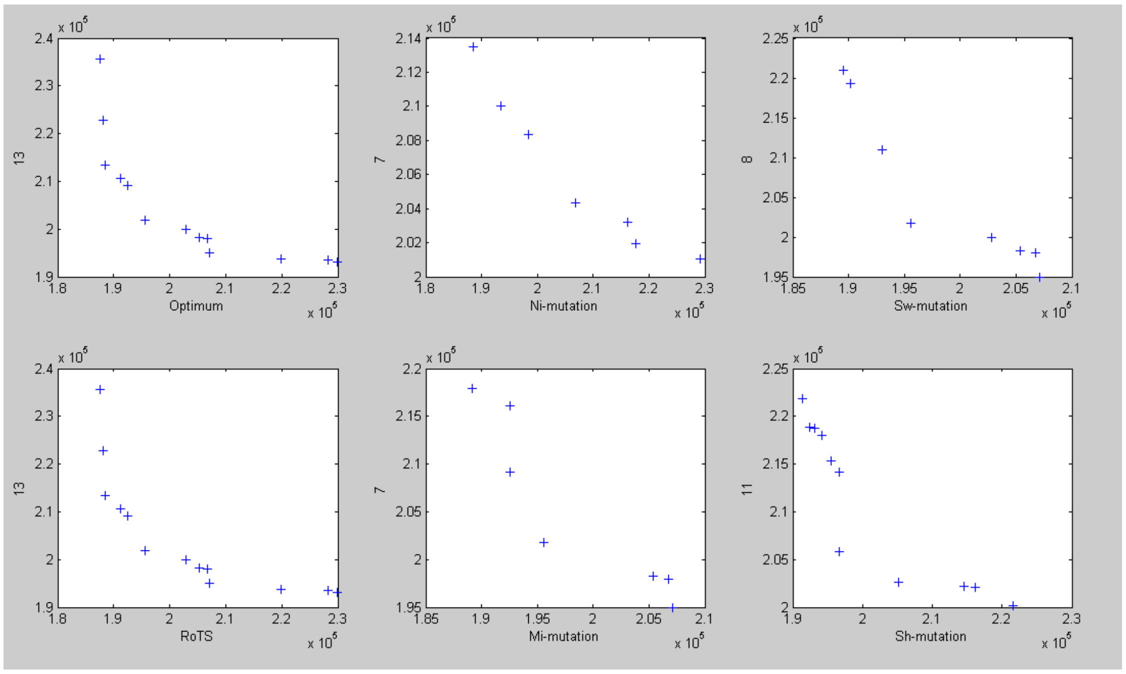


Figure 26: Pareto Front for KC10-2fl-1uni

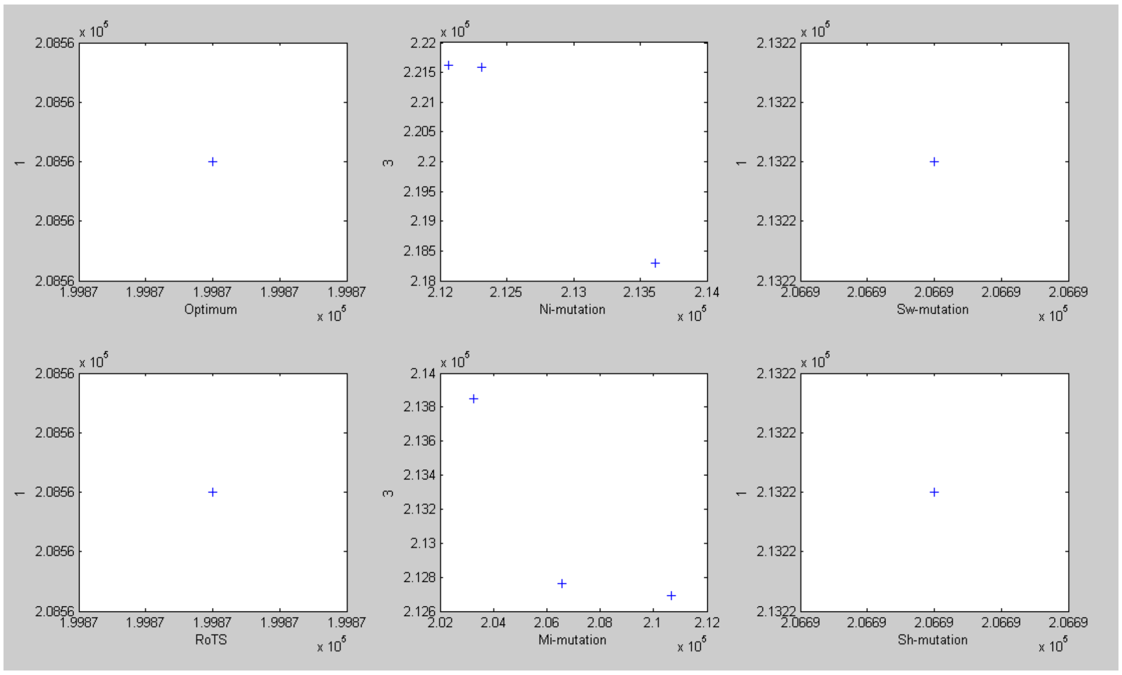


Figure 27: Pareto Front for KC10-2fl-2uni

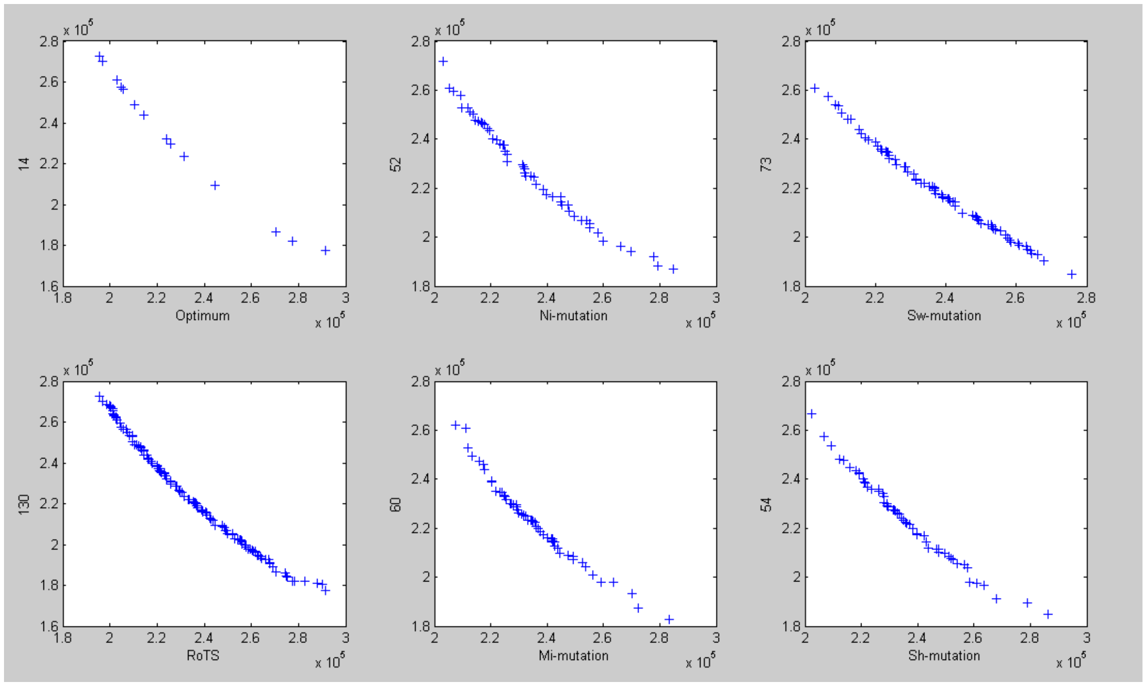


Figure 28: Pareto Front for KC10-2fl-3uni

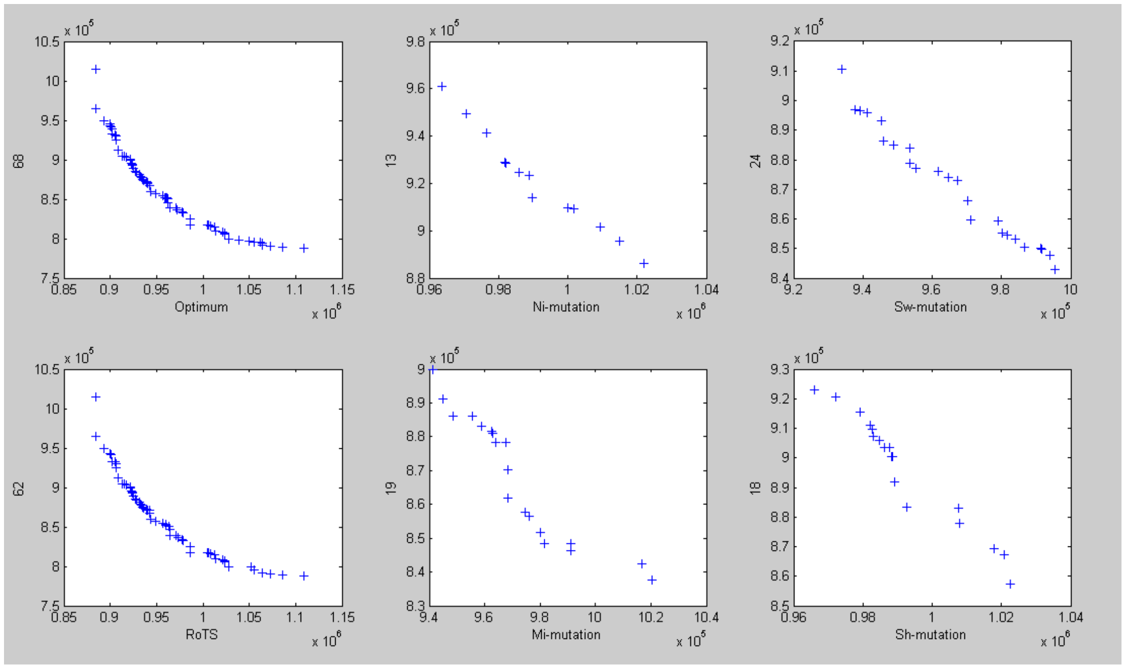


Figure 29: Pareto Front for KC20-2fl-1uni

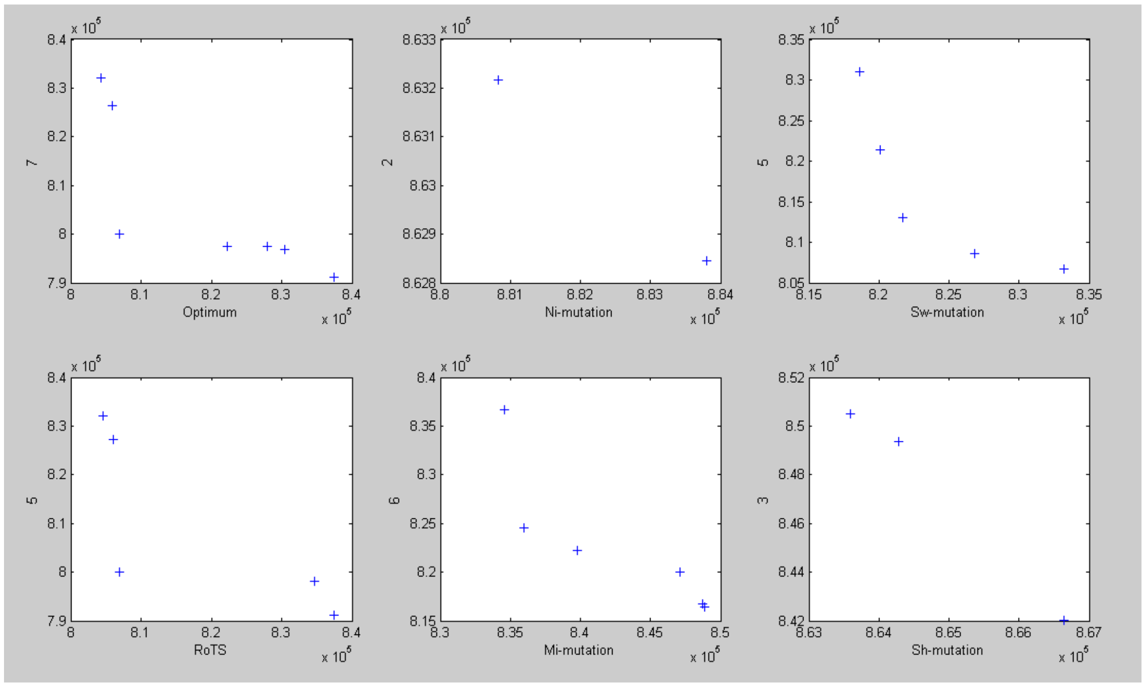


Figure 30: Pareto Front for KC20-2fl-2uni

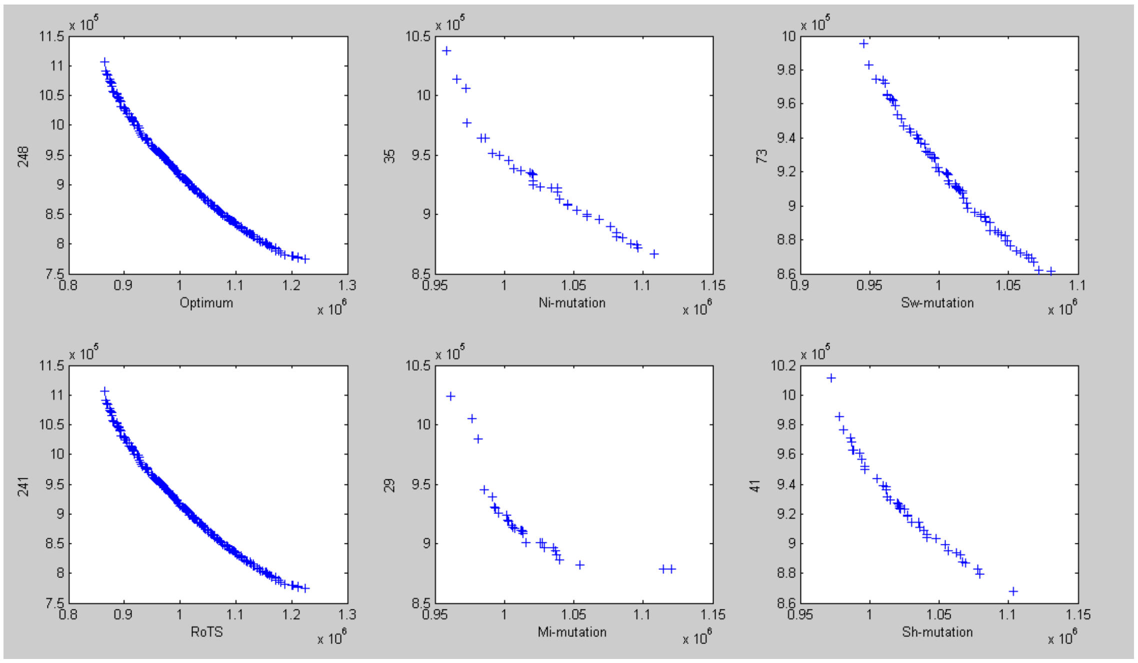


Figure 31: Pareto Front for KC20-2fl-3uni

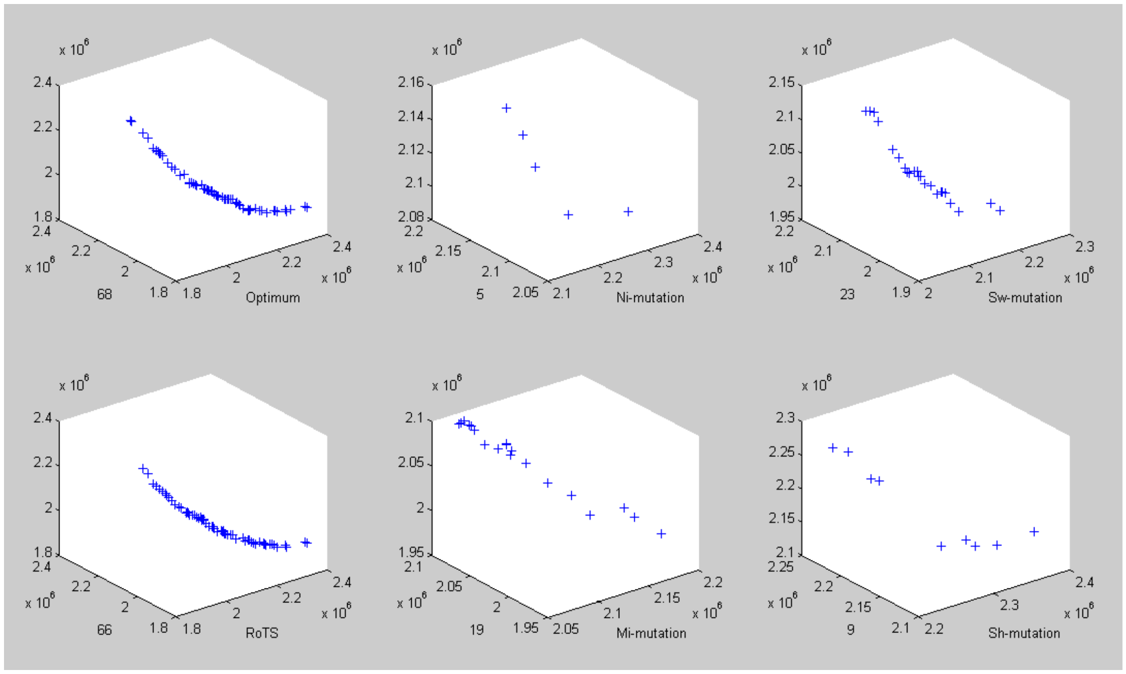


Figure 32: Pareto Front for KC30-3fl-1uni

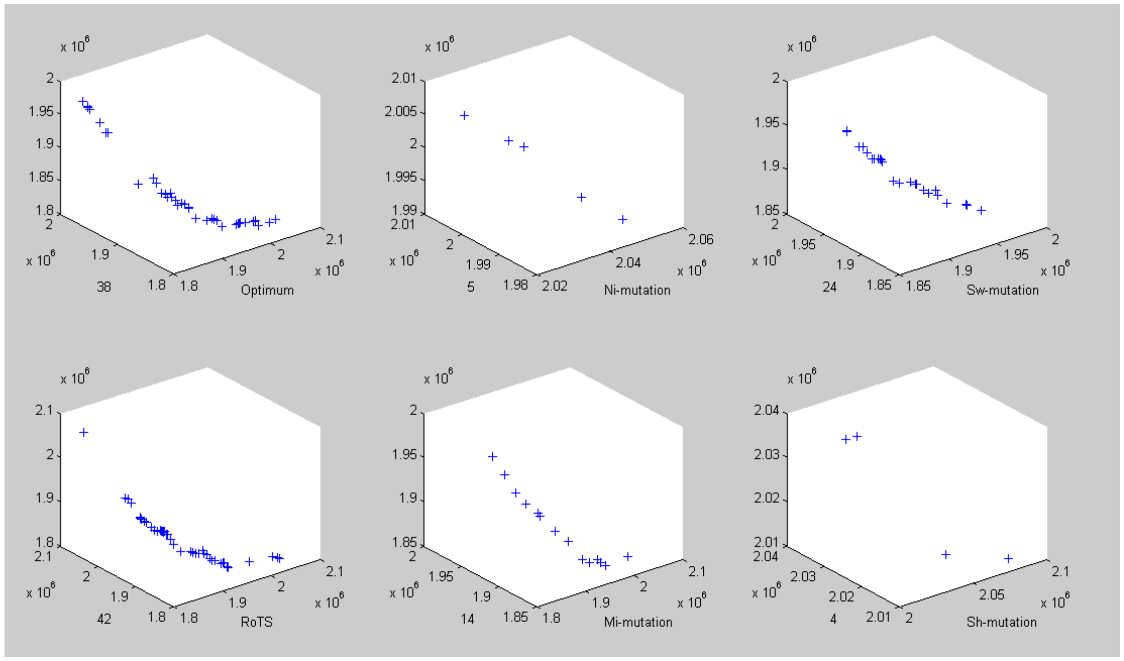


Figure 33: Pareto Front for KC30-3fl-2uni

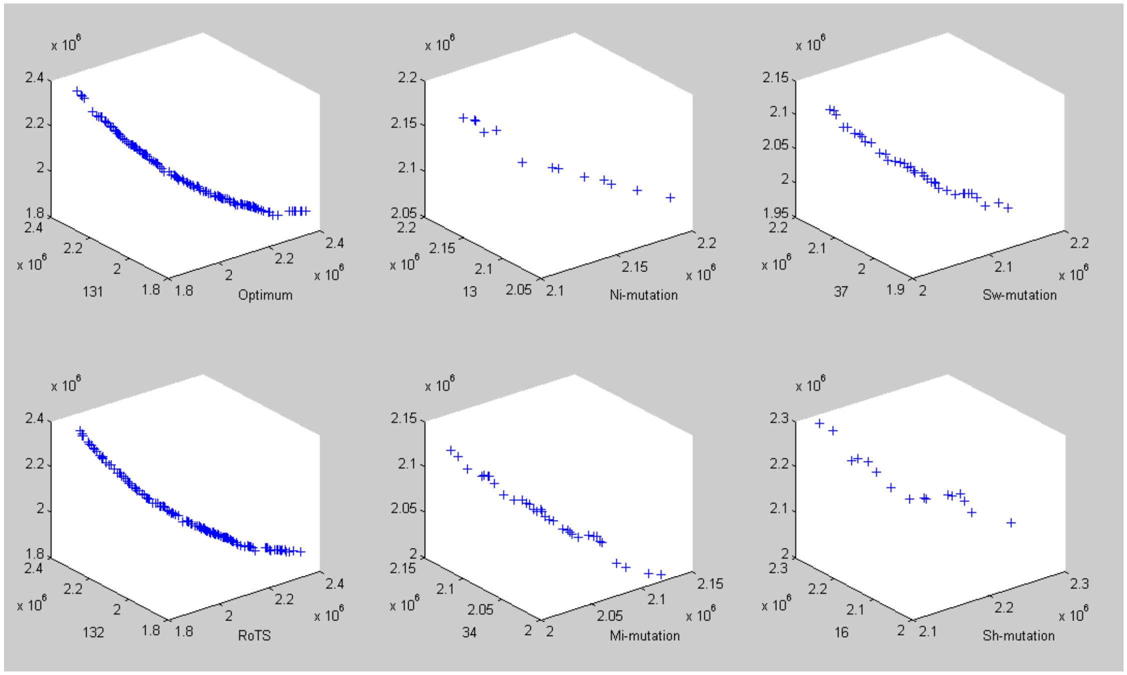


Figure 34: Pareto Front for KC30-3fl-3uni

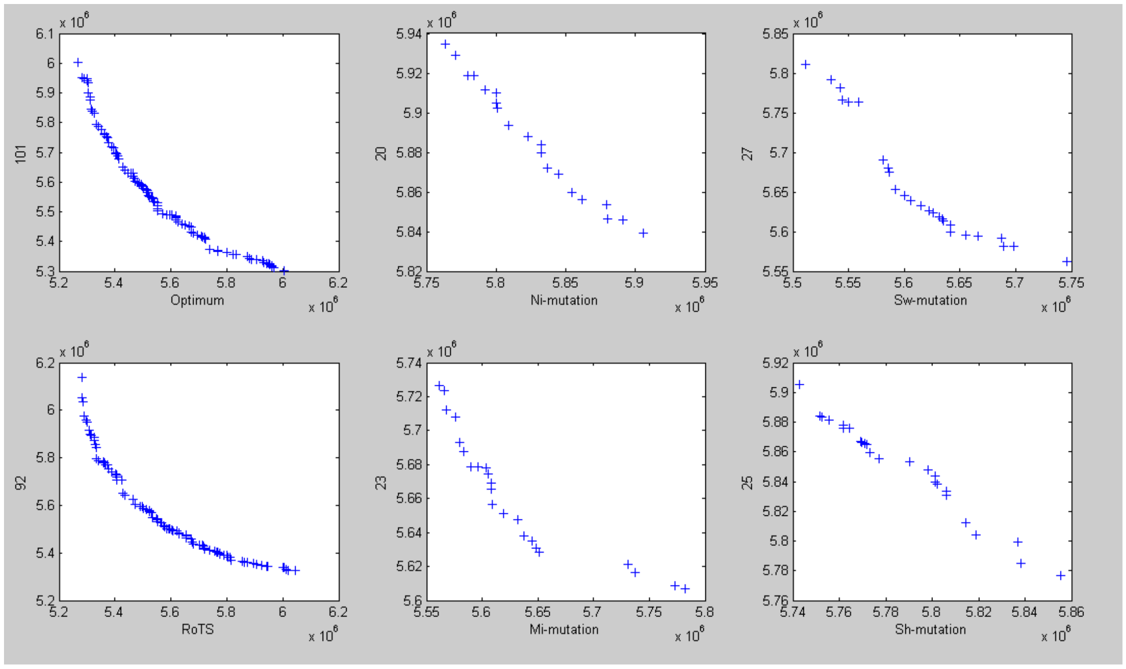


Figure 35: Pareto Front for KC50-2fl-1uni

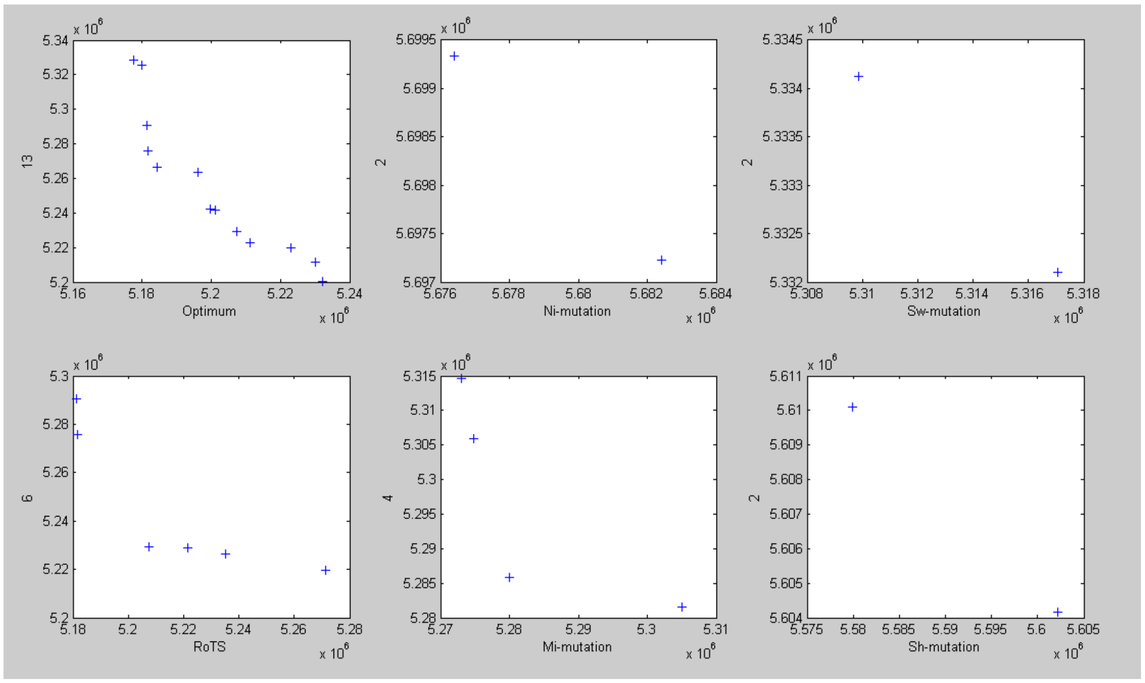


Figure 36: Pareto Front for KC50-2fl-2uni

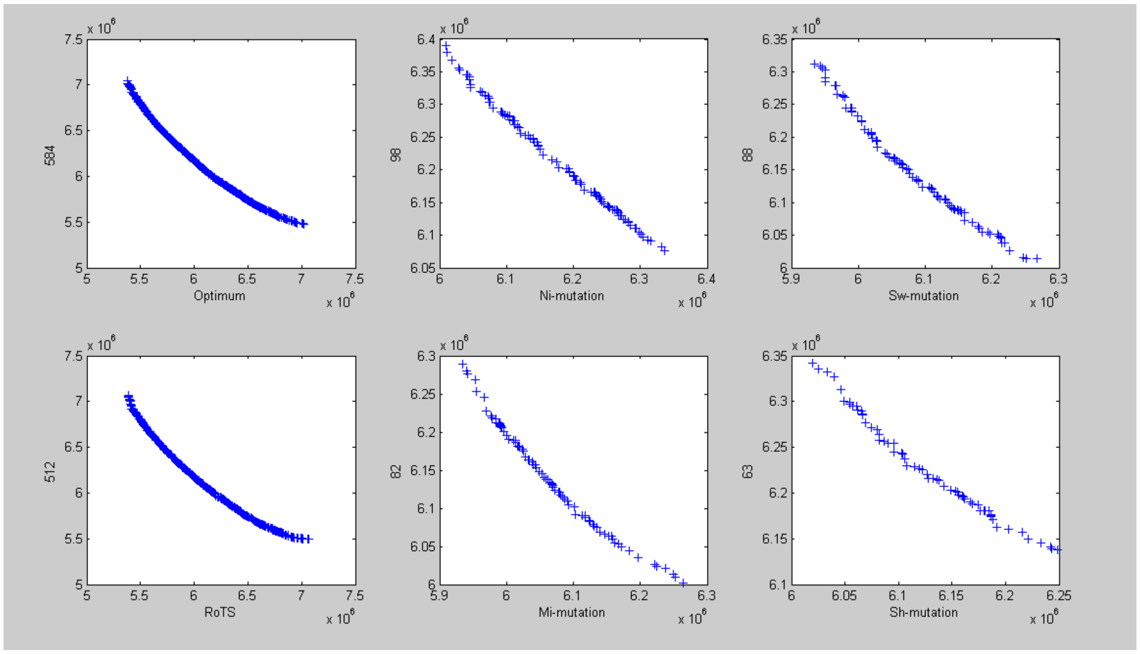


Figure 37: Pareto Front for KC50-2fl-3uni

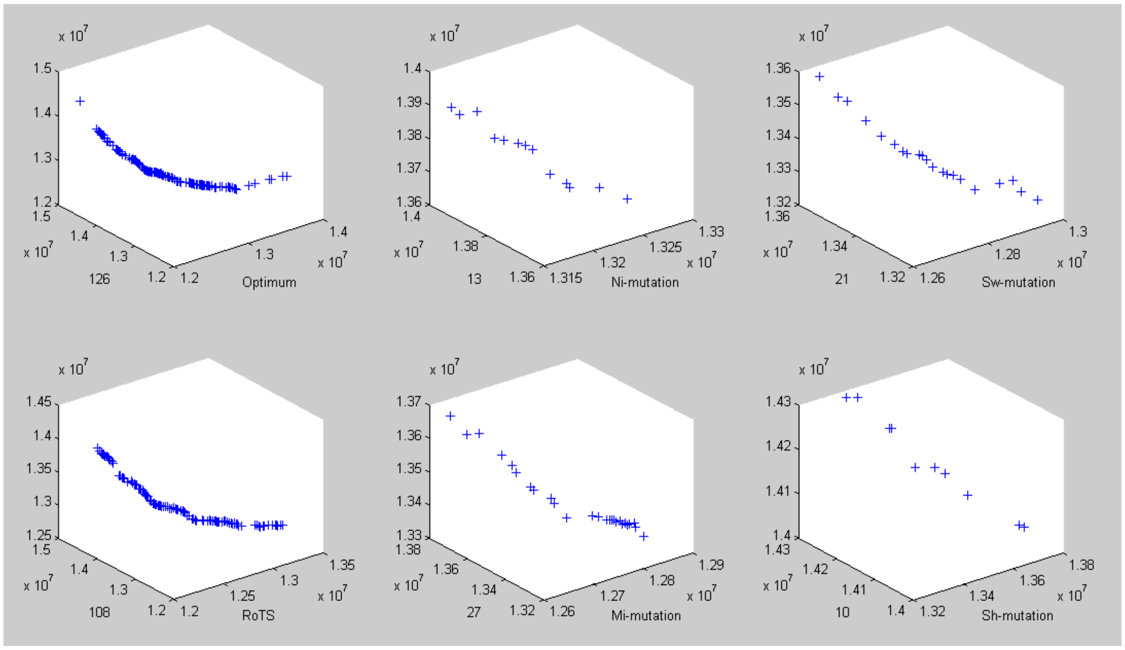


Figure 38: Pareto Front for KC75-3fl-1uni

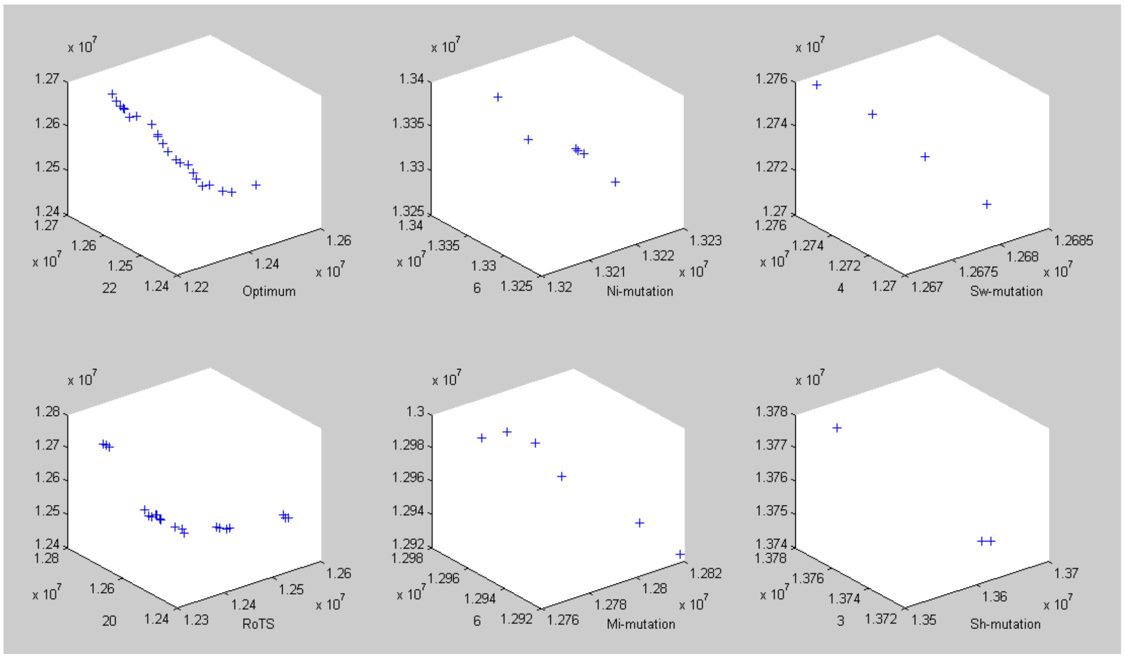


Figure 39: Pareto Front for KC75-3fl-2uni

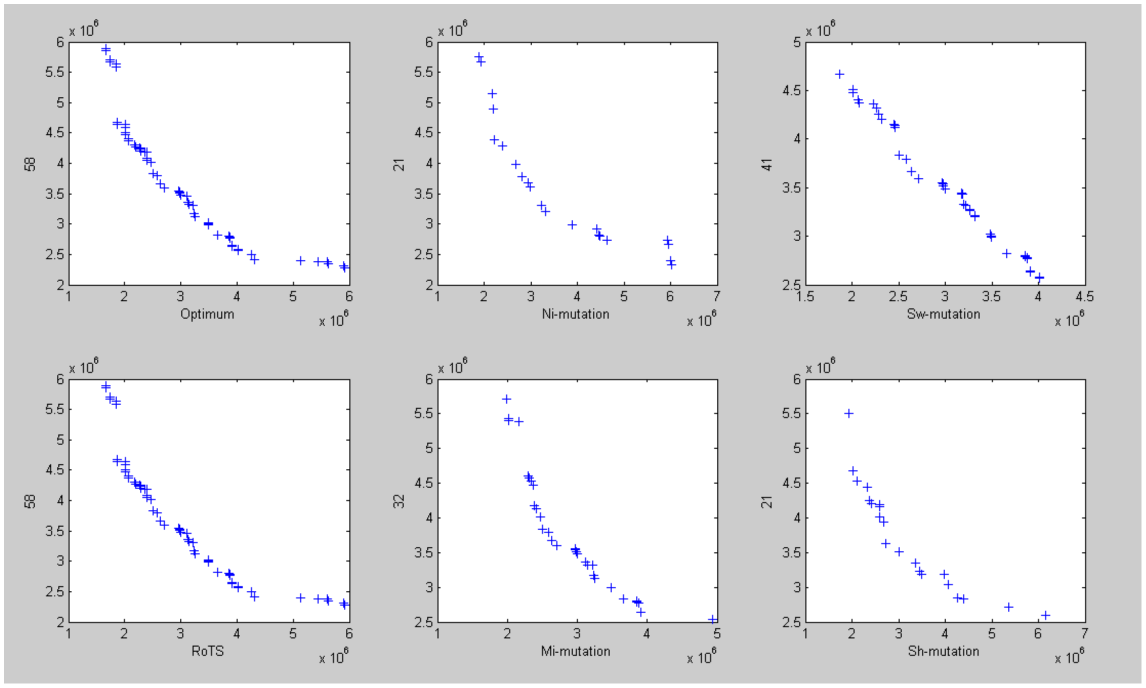


Figure 40: Pareto Front for KC10-2fl-1rl

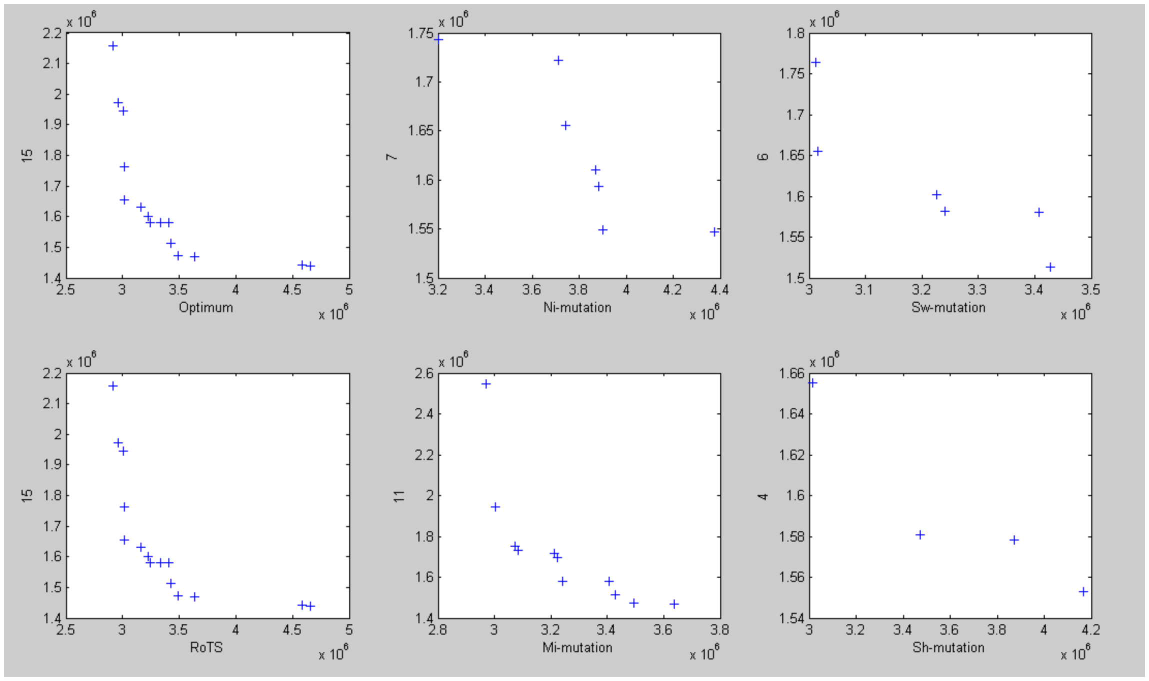


Figure 41: Pareto Front for KC10-2fl-2rl

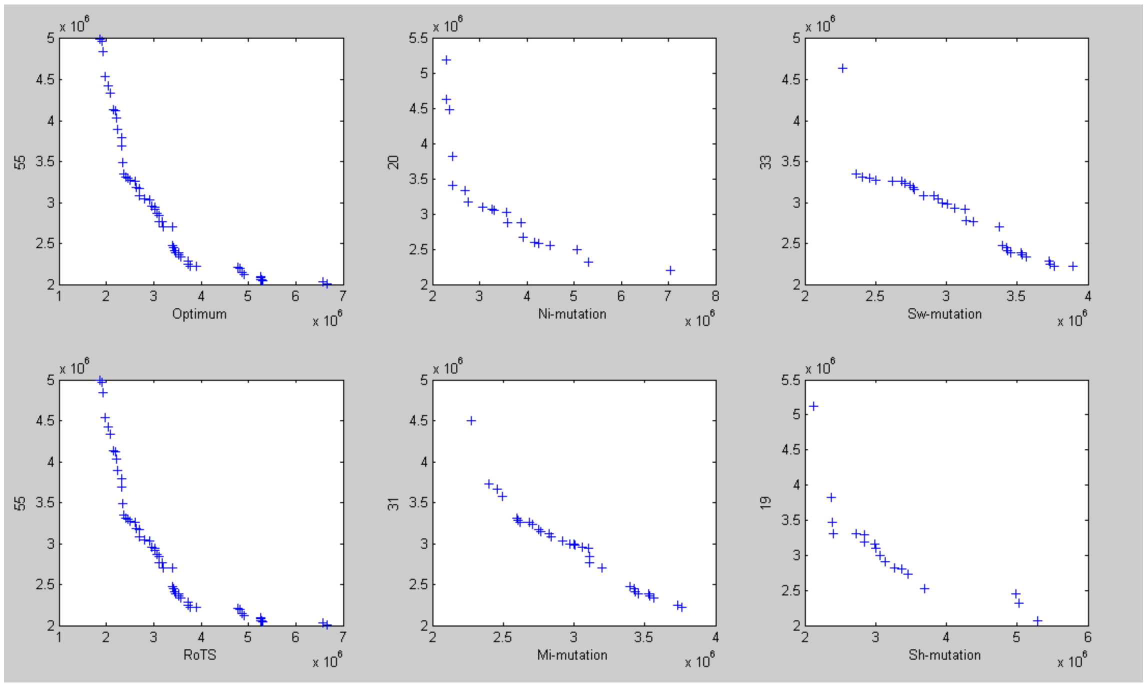


Figure 42: Pareto Front for KC10-2fl-3rl

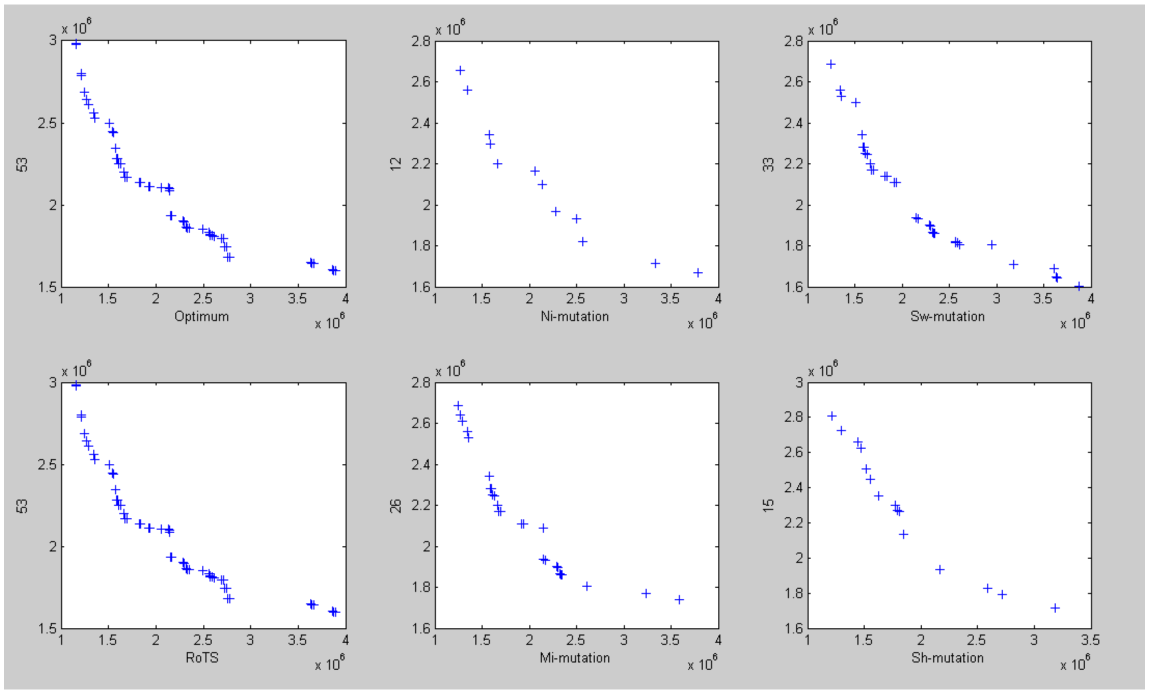


Figure 43: Pareto Front for KC10-2fl-4rl

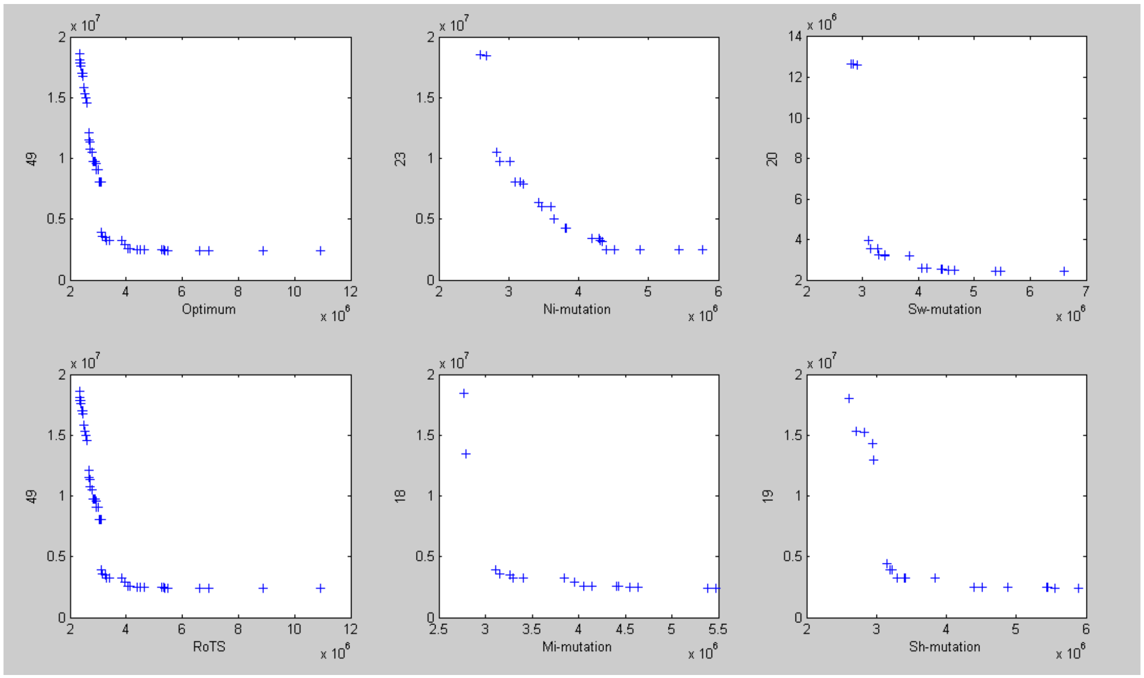


Figure 44: Pareto Front for KC10-2fl-5rl

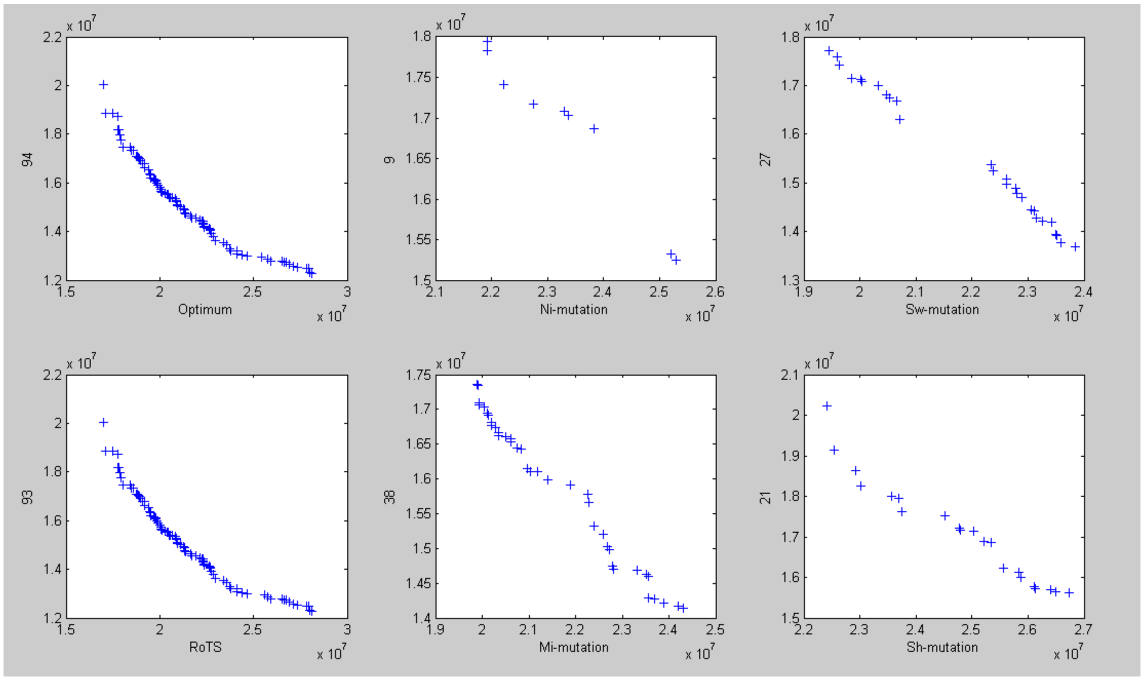


Figure 45: Pareto Front for KC20-2fl-1rl

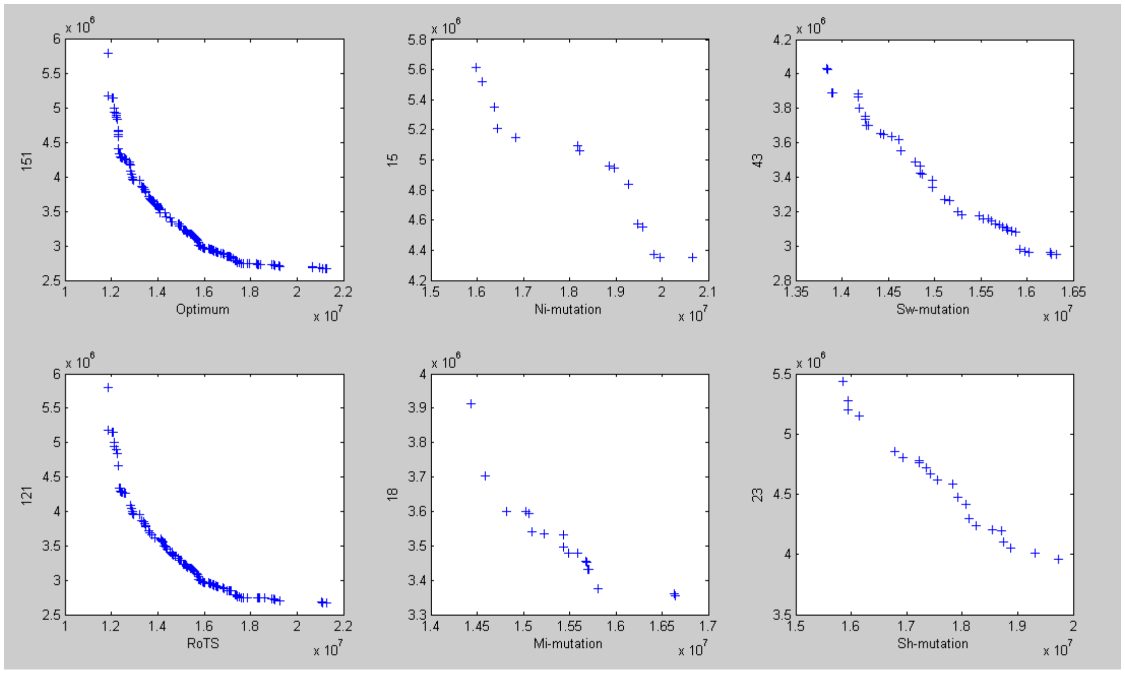


Figure 46: Pareto Front for KC20-2fl-2rl

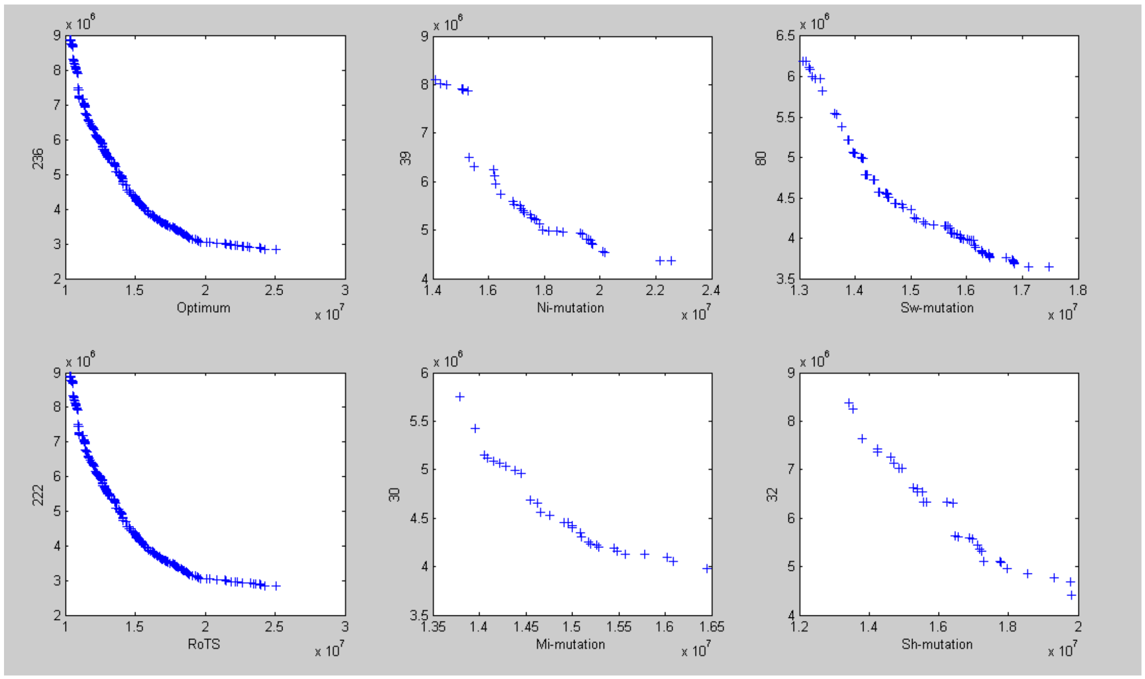


Figure 47: Pareto Front for KC20-2fl-3rl

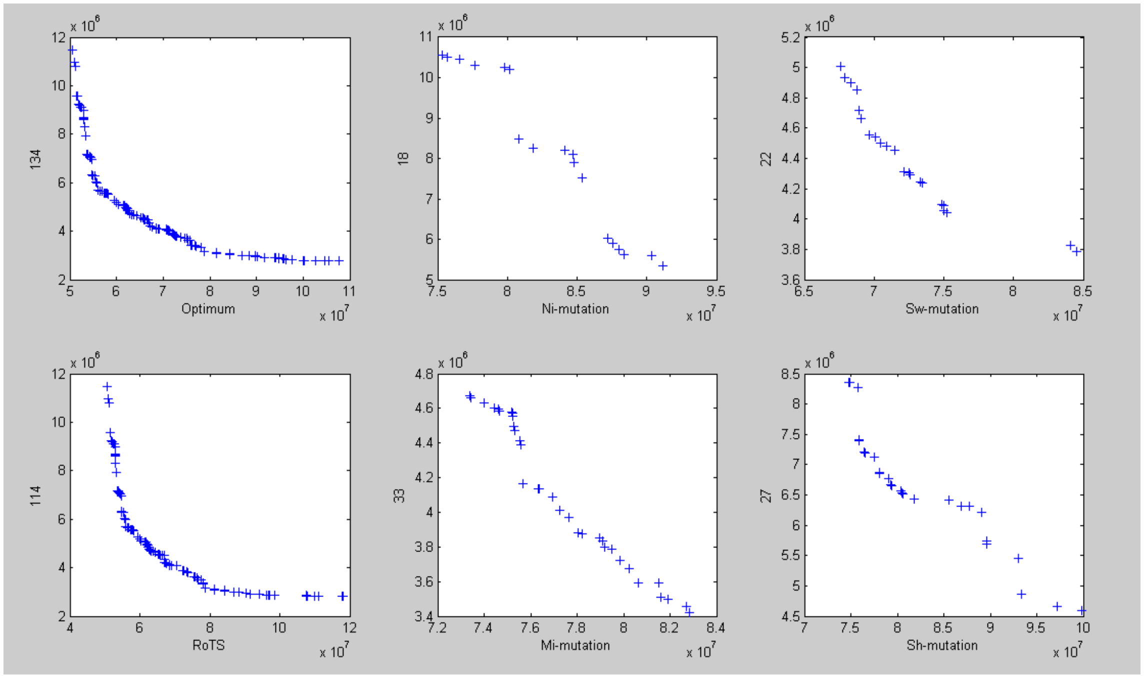


Figure 48: Pareto Front for KC20-2fl-4rl

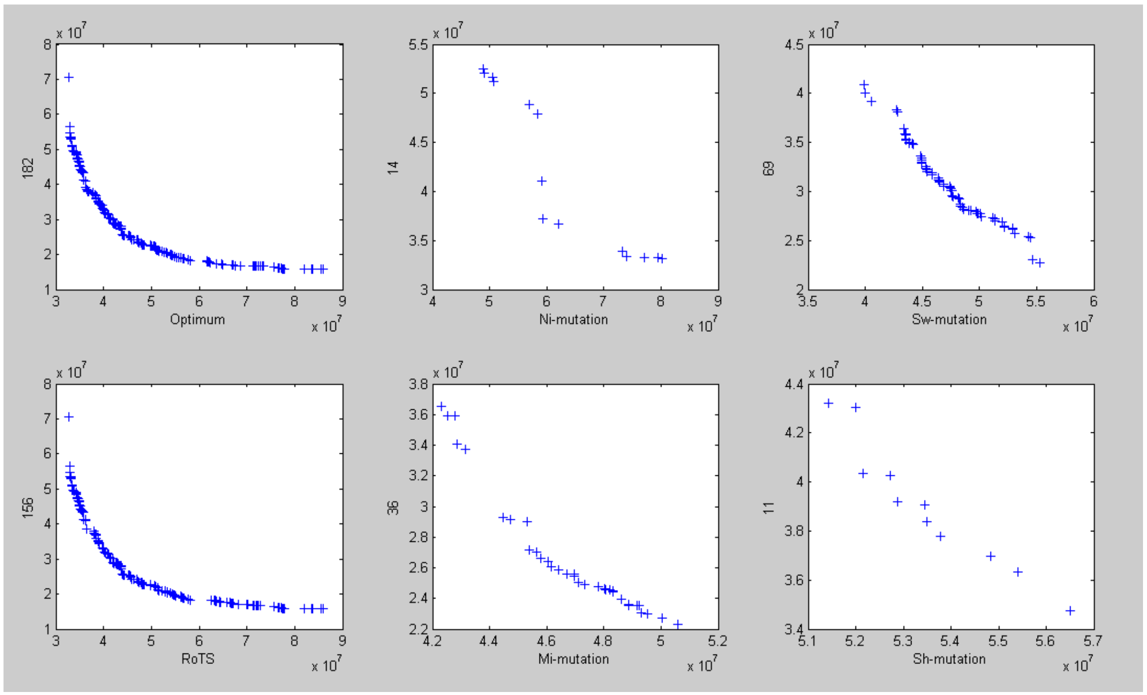


Figure 49: Pareto Front for KC20-2fl-5rl

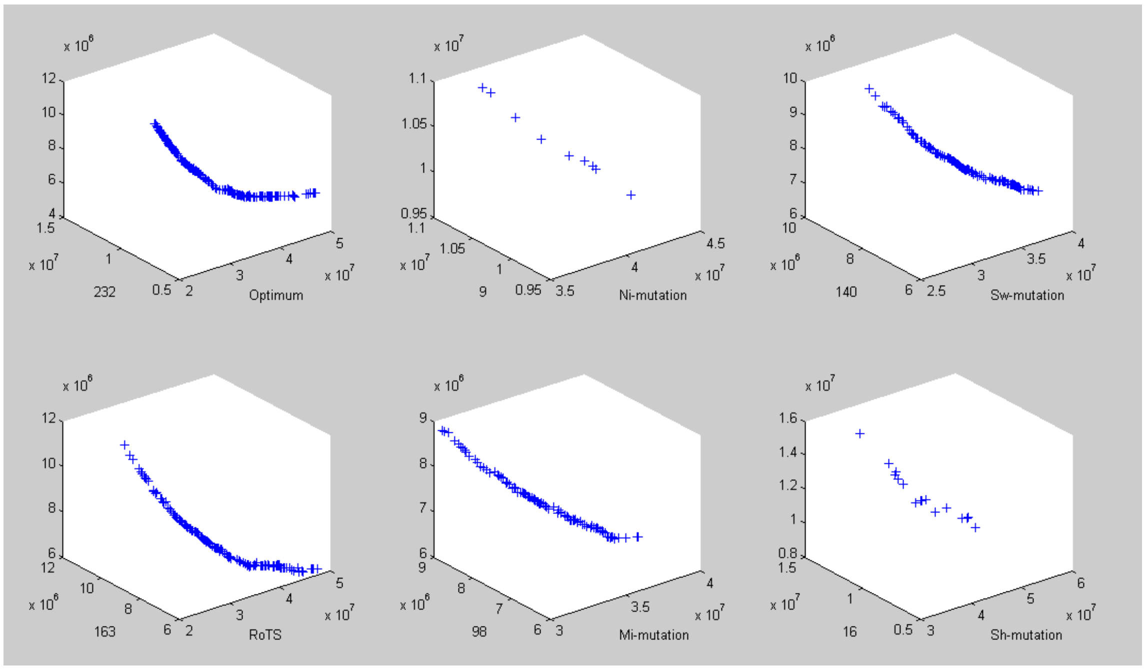


Figure 50: Pareto Front for KC30-3fl-1rl

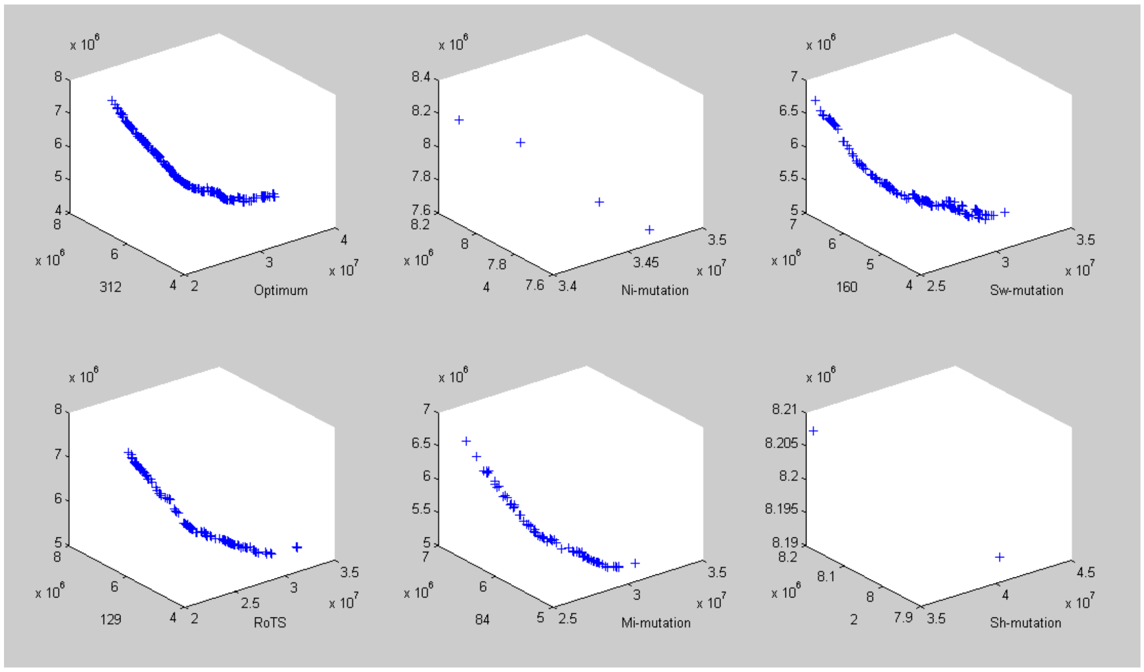


Figure 51: Pareto Front for KC30-3fl-2rl

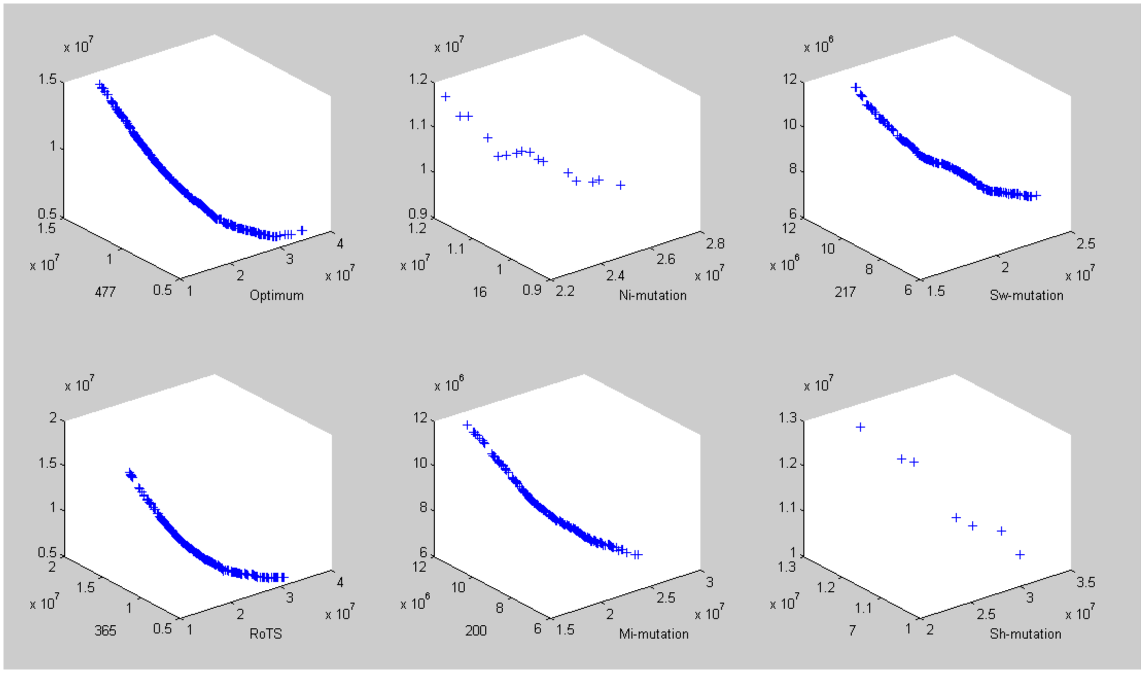


Figure 52: Pareto Front for KC30-3fl-3rl

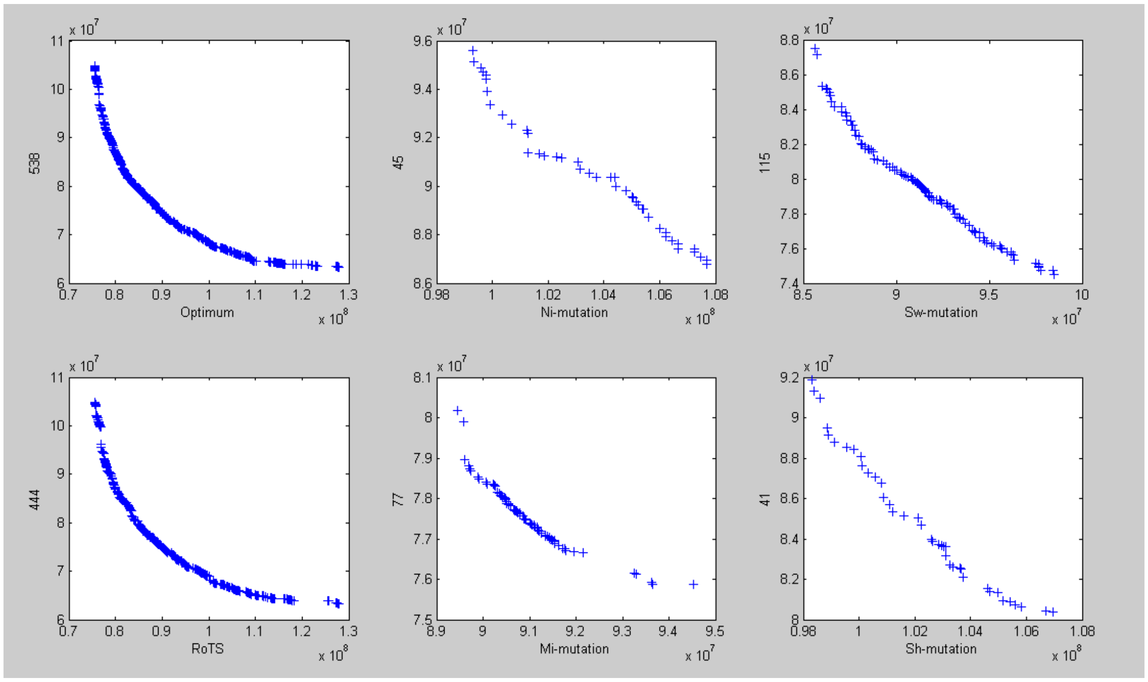


Figure 53: Pareto Front for KC50-2fl-1rl

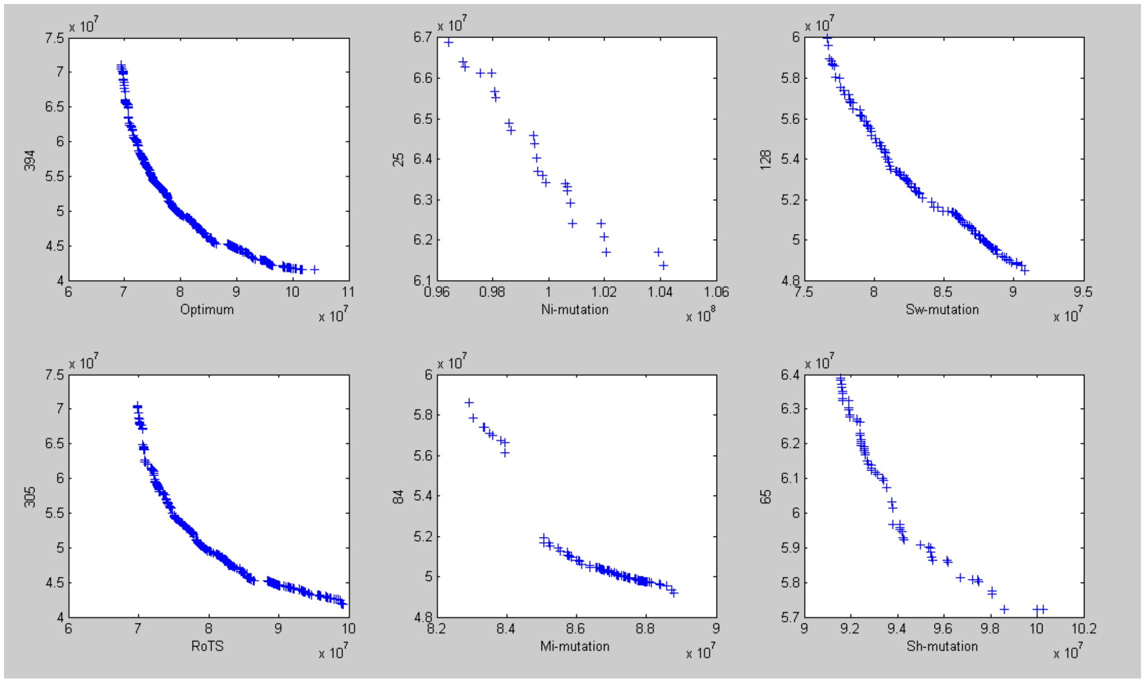


Figure 54: Pareto Front for KC50-2fl-2rl

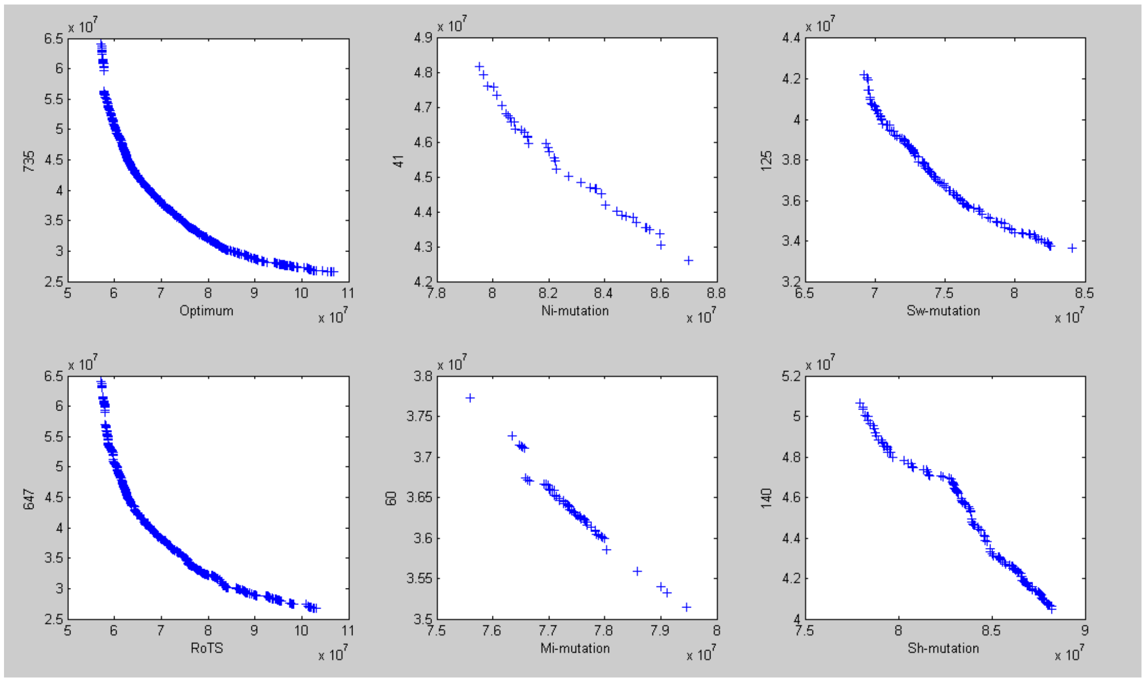


Figure 55: Pareto Front for KC50-2fl-3rl