FACILITY LAYOUT PROBLEM UNDER UNCERTAINTY

by

Hayrullah Mert Şahinkoç

B.S., Industrial Engineering, Boğaziçi University, 2012

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering

Boğaziçi University

2014

FACILITY LAYOUT PROBLEM UNDER UNCERTAINTY

APPROVED BY:

Prof. Ümit Bilge (Thesis Supervisor)	.....
Assoc. Prof. Deniz Aksen	.....
Assoc. Prof. Caner Taşkın	.....

DATE OF APPROVAL: 05.08.2014

ACKNOWLEDGEMENTS

First of all, I feel myself lucky for having opportunity to work with Prof. Ümit Bilge. I am grateful for her endless support, patience and guidance throughout my entire graduate study. This work would not be completed without her wise advices and suggestions. Secondly, I would like to express my gratitudes to Assoc. Prof. Deniz Aksen, and Assoc. Prof. Caner Taşkın for taking part in my thesis jury and providing valuable comments.

I want to express my deepest gratitude to the former and current members of BUFAIM team. I would like to thank Gökalp for his help whenever I need. I also want to thank Ezgi, Merve and Gökçe for their friendship and invaluable support during this study and for the great times we spent together.

Finally, I will never be able to find correct words to express my gratitudes to my family. I thank my mother and my father for their unlimited encouragement and giving me perspective whenever I needed. I am sure that things would be much harder if I did not have their support. Thank you for loving me more than everything you have.

I also thank TÜBİTAK for their financial support during my graduate study.

ABSTRACT

FACILITY LAYOUT PROBLEM UNDER UNCERTAINTY

The facility layout problem has usually been treated as a deterministic problem and uncertainty regarding the problem parameters has seldom been addressed. In this study, the aim is to investigate different ways of dealing with uncertainty to design a facility layout which attains robust and efficient performance under all possible scenarios. For this purpose, seven mathematical models based on the Quadratic Assignment Problem (QAP) formulation have been developed. These formulations cover alternative methodologies existing in stochastic and robust optimization literature such as minimizing maximum cost, expected cost, maximum regret and p-robustness as well as new approaches that combine them in different ways. The proposed models are solved using Genetic Algorithms (GA) incorporating operators and local improvement schemes that are specially selected and adapted for the models. As two of the models involve multiple objective functions, a Multi-Objective Evolutionary Algorithm (MOEA) has also been developed. Extensive numerical analysis enables us to compare the performance of these approaches in terms of robustness metrics and to gain important insights into ways of treating the uncertainty issue in facility layout problem.

ÖZET

BELİRSİZLİK ALTINDA TESİS YERLEŞİM PROBLEMİ

Tesis yerleşim problemi genellikle deterministik olarak ele alınmış ve problem parametrelerindeki belirsizlikler nadiren dikkate alınmıştır. Bu çalışmadaki amaç, tüm olası senaryolar altında gürbüz ve verimli performans gösteren bir tesis düzeni tasarlayabilmek için belirsizlikle başa çıkmanın farklı yollarını araştırmaktır. Bu nedenle, Kareli Atama Problemine (KAP) dayalı yedi matematiksel model geliştirilmiştir. Bu formülasyonlar, rasgele ve gürbüz eniyileme yazınlarından maksimum maliyet, beklenen maliyet, maksimum pişmanlık, p-gürbüzlük gibi alternatif yöntemleri ve bunların yanı sıra bu yöntemleri farklı şekillerde birleştirmek isteyen yeni yaklaşımları içermektedir. Önerilen modeller, özel olarak seçilen ve uyarlanan operatörler ve yerel geliştirme programları içeren Genetik Algoritma (GA) kullanılarak çözülmüştür. Modellerin iki tanesi çok amaçlı amaç fonksiyonları içerdikleri için, bir Çok Amaçlı Evrimsel Algoritma (ÇAEA) da geliştirilmiştir. Kapsamlı sayısal analiz, bu yaklaşımların performanslarını farklı gürbüzlük ölçütleri bazında karşılaştırmak ve tesis düzenlemesi probleminde belirsizlik sorununun tedavi yollarını kavrayabilmek için bize fayda sağlamaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$bestSolution$	Best solution obtained so far in single objective algorithms
C_{max}	Maximum travel distance across all scenarios
c_s^*	Cost of optimal solution of scenario s
d_{kl}	Distance between locations k and l
$expectedcost$	Expected cost of a solution
f_{ij}^s	Flow between departments i and j under scenario s
f_{ij}^*	Weighted average of flows between departments i and j across all scenarios
$fitness^t$	Fitness of a solution in iteration t
$immigrationRate$	Possibility that a solution enter immigration procedure
$iterationCount$	Current number of iteration in the genetic algorithm
$maxIteration$	Number of iteration that genetic algorithm terminates.
$migrateCount$	Number of solutions enter immigration procedure in every iteration
$multiplier$	Multiplier for the penalty coefficient
p	Desired robustness level
p^s	Desired robustness level for scenario s
p_s	Probability that scenario s occurs
p_{coef}^t	Penalizing multiplier at iteration t
$p_{coef}^{initial}$	Penalizing multiplier at the beginning of the algorithm
p_{coef}^{final}	Penalizing multiplier at the end of the algorithm
P	Current population in the genetic algorithm
P_{chi}	Child population in the genetic algorithm
P_{com}	Combined population in the genetic algorithm
P_{mig}	Immigrant population in the genetic algorithm
P_{par}	Parent population in the genetic algorithm
P_{path}	Path population in the crossover operator
$populationHistory$	Population history matrix
$populationSize$	Number of solutions in the population

$parentSize$	Number of solutions in the parent population
R^{abs}	Absolute regret
R^{abs}_{max}	Maximum absolute regret across all scenarios
R^{rel}	Relative regret
R^{rel}_{max}	Maximum relative regret across all scenarios
$violation$	Violation of a solution in the objective function
x_{ik}	Binary variable which takes value of 1 if department i is located at location k, and 0 otherwise

LIST OF ACRONYMS/ABBREVIATIONS

GA	Genetic Algorithm
MOEA	Multi-objective Evolutionary Algorithm
MOP	Multi-objective Optimization Problem
NSGA-II	Non-dominated Sorting Genetic Algorithm
QAP	Quadratic Assignment Problem
TS	Tabu Search

1. INTRODUCTION

Facility layout decisions are strategically important and difficult to reverse as their monetary cost consequences may be huge. The effect of the related decisions lasts for a long time horizon and the environment in which the facility operates might change drastically. Decision maker has to deal with variability occurring in product mix and demand quantities.

In spite of this inherent imprecision, the facility layout problem has usually been treated as a static deterministic problem. The typical design objective is the minimization of material handling cost, which is calculated from the product of flow densities between departments and the distances between candidate locations. This problem is often formulated as a Quadratic Assignment Problem (QAP) which is also selected as the facility layout benchmark problem in this thesis.

QAP is one of the most difficult problems in the NP-hard class (Sahni and Gonzales, 1976) and it models many real-life problems in several of facilities design and facilities location problems. During the past decades, several exact and heuristic algorithms have been developed for solving this problem. However, uncertainty regarding problem parameters has seldom been addressed even though the design parameters are highly exposed to fluctuations during the long time horizon when layout decisions are in effect.

Uncertainty can be caused by the lack of accuracy of the input data due to poor measurements or external factors that have severe effects on the quality of layout design. In our case, uncertainty and variability might occur in product mix, part routing and process plans regarding the production amounts and these are all represented in the flow matrix of the QAP formulation.

Optimization under uncertainty is usually handled as either stochastic optimiza-

tion or robust optimization. The difference between these two approaches is mainly based on the fact that in stochastic optimization, uncertain parameters are governed by probability distributions that are known in advance by the decision maker where these probability distributions are not necessary or in some cases not available for robust optimization. In robust optimization, the common attempt is to optimize the worst-case performance of the system (Snyder, 2006) and the objective is to solve the optimization problem not only for the nominal solutions but also for the robust optimal solutions.

The uncertain parameters in both stochastic and robust optimization may be either continuous or discrete where discrete parameters are defined by using scenario approach. The scenario approach results in more manageable models and most robustness measures do not require the decision maker to estimate scenario probabilities. In this thesis, we implement the scenario approach.

The aim of this thesis is to investigate how the inaccuracy of input data affects the layout design objective in order to design a facility which attains robust and efficient performance under any possible realization of the uncertain parameters. As the definition of performing well varies from application to application choosing appropriate performance measure is a significant part of the modeling process. For this purpose, several QAP based mathematical formulations have been developed. These formulations cover alternative methodologies existing in stochastic and robust optimization literature; such as minimizing expected cost, maximum cost, maximum absolute and relative regret and p-robustness where the first four formulations concentrate on different performance measure and the last approach aims to combine expected cost and regret concepts.

Furthermore, two multi-objective formulations having different viewpoints are developed pursing the goal of generating solutions that behaves well in many aspects and concerns. In the first multi-objective formulation, two of the performance measures are used together. In the second multi-objective formulation, a totally different approach is

taken and minimizing the cost of each scenario becomes a separate objective. This is a natural approach as pointed out by Aissi *et al.* (2009), however rarely considered in the robust optimization literature. So our last formulation dwells on the fact that robust solutions should be in the Pareto front of a scenario-based multi-objective approach and investigates the relationship among robust optimization and the multi-objective version. Furthermore, Iancu and Trichakis (2013) remark that worst-case minimization in robust optimization might lead to "un-optimized" decisions for the non-worst-case scenarios and we take this into account in our solution procedures.

Proposed mathematical QAP based formulations are solved using a genetic algorithm (GA) to search the optimal or Pareto optimal solutions for our formulations. Operator and local improvement schemes are specially selected and adapted for our formulations. For the multi-objective formulations, we proposed a multi-objective genetic algorithm to approximate an efficient frontier in reasonable computation times. The multi-objective genetic algorithm used in this thesis is developed after investigating the vast wide approaches in multi-objective evolutionary algorithm (MOEA) literature and is similar to Non-dominated Sorting Genetic Algorithm-II (NSGA-II) proposed by Deb *et al.* (2002). Nevertheless, there are some modifications that address the specific requirements of our formulations.

Extensive numerical analysis enables us to compare the performance of these approaches in terms of robustness metrics and to gain important insights into ways of treating the uncertainty issue in facility layout problem.

The rest of this thesis is organized as follows. Chapter 2 presents a literature review on stochastic and robust optimization, the QAP, GA, multi-objective optimization and multi-objective genetic algorithms. Chapter 3 presents the seven proposed formulations based on the QAP in detail. In Chapter 4, both single and multi-objective genetic algorithms developed are explained. Chapter 5 covers the numerical experiments and comparisons of proposed methods. In the last chapter, Chapter 6, concluding main findings are pointed and suggestions for future research are provided.

2. LITERATURE REVIEW

This chapter provides a brief literature survey about the relevant topics with this thesis. In Section 2.1, facility layout problems under uncertain environment in literature and classifications of robust optimization methodology and terminology are discussed. Then, Section 2.2 outlines QAP terminology with the solution techniques referred. At the end of this chapter, multi-objective optimization literature is covered in Section 2.3.

2.1. Stochastic and Robust Optimization

Facility layout decisions can be costly and it is difficult to reverse them thus their impact spans for a long time. However, during this time, some of the parameters may change rigorously. As a consequence, dealing with uncertainty in layout problems are very important and various kinds of stochastic and robust optimization performance measures aim to achieve this goal. To gain knowledge on both optimization paradigms, books on both methodologies are provided. In Kouvelis and Yu (1997), a framework for the robust discrete optimization is presented. The efficiency and expected performance of robust solutions are discussed. Complexity of the computational results for the robust formulations of various well-known problems including production planning problem and assignment problem are illustrated and some solution techniques for these problems are discussed.

Ben-Tal *et al.* (2009) explain the importance of handling with data uncertainty in optimization and indicates this can affect the quality of the solutions severely. They focus on modeling the robust counterparts of linear programming problems and extend their finding to more general optimization problems.

A comprehensive survey presented by Beyer and Sendhoff (2007) also review the current methodology with a detailed investigation on the different approaches in robust

optimization. It is stated that the main focus in robust optimization researches are the methods for developing mathematical programming, creating robust counterparts of nonlinear problems and in the evolutionary algorithms.

For the stochastic programming problems, Shapiro and Dentcheva (2009) present the modeling and theory of stochastic programming models with concentrating on its foundations and presenting recent advances in the chosen areas. Modeling issues occur in stochastic programming are explained in the context of specific models.

There are several articles discussing the uncertainty issue in layout design problems. The importance of nondeterministic models in the layout problems are also covered using robust optimization methodology. The notion of robustness and the robust optimization methodology has been studied by many articles and numerous different techniques have been applied.

Benjaafar and Sheikhzadeh (2000) analyze facility layout designs under stochastic environments and develop generalized models of QAP each based on different layout design methodologies and includes additional allocation and department opening decisions. Their models also involve robustness constraints and their evaluation for the performances of different types of plant layout alternatives also involves robustness concerns although, the term flexibility is used instead of robustness terminology.

Norman and Smith (2006) present a formulation for the unequal area facility block layout problem that takes uncertainty into considering in the problem parameters where these stochastic uncertain parameters are continuous with predefined probability distributions.

To the best of our knowledge, robustness notion is first mentioned in a facility design problem by Rosenblatt and Lee (1987). The model chosen to illustrate the facility layout problem is QAP although only numerical analysis are shown without mentioning the formulation. The uncertainty is caused by the fluctuations in the

amount of produced items and all represented in the flow matrix with different level each represented under a separate scenario. They have introduced the robustness approach in a primitive manner and indicating that a solution is considered robust if it performs well under as many as scenarios possible.

Kouvelis *et al.* (1992) state the importance of designing a robust facility layout instead of just minimizing the expected material handling cost and define robustness as being close to the optimal solution for a wide variety of demand scenarios. They indicate that it would be unnecessary for a solution to be optimal under any specific demand scenario and emphasize that knowing the probability of realization of each scenario is unnecessary in robustness approaches which is the foundation of the p-robustness approaches in prosecuting researches. They also extent their results for the multi-period layout problem.

Snyder (2006) presents a comprehensive survey on the uncertainty treatments in facility location problems in which almost every stochastic and robust performance measures are stated and explained briefly. Also examples from more general logistic problems are covered to illustrate different types of stochastic robust optimization approaches.

There are various approaches that employ different concepts existing in robust optimization literature. One of most applied approaches includes minimizing the cost of the scenario that the maximum cost occurs and minimizing the maximum regret occurred across all scenarios. The main attraction of the minimax measures is that it is not required to estimate the scenario probabilities. A similar approach includes minimizing the maximum regret where the regret for a scenario is the deviation from the optimal solution for that scenario. Aissi *et al.* (2009) present a survey on the topic of discrete minimax cost and minimax regret versions of the combinatorial optimizations problems The minimax regret is covered with interval objective function coefficients, in Mausser and Laguna (1999a) for linear programs, in Mausser and Laguna (1998) for mixed integer formulations, and in Józefczyk and Siepak (2013) for scheduling

problems. Heuristic solution techniques to solve minimax problems are in Mausser and Laguna (1999b) for general mixed integer problems and in Serra and Marianov (1998) for p-median problem.

Other widely applied techniques also exist such as p-robustness enlightened by Snyder and Daskin (2006). The concept of p-robustness is clearly defined and the aim of combining two measure of minimizing expected cost and minimizing worst-case regret by placing the constraints related with the latter and putting the former in the objective function is illustrated in a facility location problem. The same methodology is named as γ-robustness in Lim and Sonmez (2013) where the facility relocation problem is studied. In Liu *et al.* (2010) and in Peng *et al.* (2011) similar work is done for modeling capacitated network design model.

Other examples for robust optimization approaches are α-reliable approaches studied by Daskin *et al.* (1997) and Chen *et al.* (2006) both for facility location modeling and robust optimization for calculating conditional value at risk in Quaranta and Zaffaroni (2008). These techniques include introducing some boundaries to maintain the robust features in the feasible solution space.

2.2. Quadratic Assignment Problem (QAP) and Genetic Algorithm (GA)

QAP is one of the most difficult problems in the NP-hard class (Sahni and Gonzales, 1976). In Loiola *et al.* (2007), a comprehensive survey about QAP is presented stating that many real-life problems in several areas such as; facilities location, combinatorial data analysis are modeled as QAP. Some of the most important QAP formulations are stated and classified and a detailed discussion is made on the exact and heuristic solution techniques, including metaheuristic strategies. In addition, the main research trends and tendencies in the QAP literature are identified to guide future researches and these material constructed a foundation for our QAP investigation.

In Zhang *et al.* (2013), general purpose mixed integer linear programming solvers

are addressed to solve QAP. Different types of formulations are obtained by using linearization techniques to find tight lower bounds and efficient performance to solve test problems in reasonable times. In Resende *et al.* (1995) an interior point algorithm for linear programming is developed to compute lower bounds for the QAP in order to serve branch-and bound techniques to come up with optimal solutions with a convenient effort. However, all these techniques are proved to be impractical for all but small-sized problems. As QAP is computationally NP-hard, large problem instances can require a great amount of time to be solved optimally by exact methods. The computational difficulty in solving QAP motivated the development of many heuristic algorithms.

During the recent decades, almost every heuristic search technique has been adopted to QAP. Pardalos and Resende (1994) has produced a greedy randomized adaptive search, called GRASP. They claim that their solution technique is capable of quickly producing good quality solutions for not only QAP but also a wide variety of combinatorial optimization problems.

On the other hand, in many of the researches, metaheuristic techniques especially genetic algorithms, are employed to solve QAP. Taillard (1995) compared the performances of three different tabu search algorithms and a hybrid genetic algorithm, indicating that all these solution methods are efficient heuristics and none of among them outperforms the other.

Tate and Smith (1995) are among the pioneers that consult genetic algorithms to solve QAPs. The all main stages of the genetic algorithms including the tuning for the parameters are investigated and they draw the conclusion that GAs are powerful tools to overcome with the combinatorial problem like QAP.

Drezner (2005) proposes a compounded genetic algorithm consisting of two phases: where the first phase evolves good solutions of the second phase. His algorithm is hybridized with tabu search and successful results are obtained for the test problems in QAPLIB. Later on, in Drezner (2008) some modifications mainly on diversification

concerns are proposed and better results are obtained.

Misevicius (2003) proposes a genetic algorithm hybridized with the improvement procedure called ruin and recreate procedure. In Misevicius (2004), a hybrid genetic algorithm using some elements of tabu search at its local improvement stage, is proposed.

Ahuja *et al.* (2000) propose a greedy genetic algorithm as it includes many greedy principles on both intensification and diversification concerns. They also present several alternatives for crossover operator, concepts of immigration and tournament and a detailed fine tuning for the algorithm parameters. The methodology and the operator schemes of this paper serves as a basis for the genetic algorithms developed in this thesis.

Knowles and Corne (2002) present a multi-objective version of QAP (mQAP) considering several flow and distance matrices. The authors state that mQAP can be useful for some layout problems, such as hospital layout where different types of flows; i.e. doctors, patients, nurses lead to different objectives. They investigate landscape analysis issues for approximating Pareto front using mQAP as a benchmark. In their paper, a hybrid local search algorithm is presented to approximate the Pareto front and in Knowles and Corne (2003) they formulate some instance generators and test suites for mQAP.

In Paquete and Stützle (2006), a two-phased local search procedure is followed for solving the bi-objective QAP and ant colony optimization (ACO) metaheuristic is employed for again the bi-objective QAP in López-Ibánez *et al.* (2004). There are some works applying multi-objective evolutionary algorithms are employed for mQAP; such as Kleeman *et al.* (2004) and Day and Lamont (2005) where some variations of multi-objective messy genetic algorithm is used.

2.3. Multi-objective Optimization and Multi-objective Evolutionary Algorithms

In multi-objective optimization problems (MOP), the goal is to optimize k objective functions simultaneously (Coello *et al.*, 2007). The objective function value for each feasible solution becomes a vector function whose elements represents each objective function. In MOPs, two Euclidean spaces are considered: N-dimensional space of the decision variables and K-dimensional space of the objective functions. A point in the former space represents a solution and gives a specific point in the second space, which displays the quality of this solution with regards to the objective function values.

The concept of optimality changes in MOPs and the goal is to find good compromises (or trade-offs) among objectives rather than searching for a single global optimal point (Marler and Arora, 2004). Although, single objective optimization problems might have a unique optimal solution, MOPs present a set of solutions which produce vectors whose components represent trade-off in objective space. A decision maker then chooses an appropriate solution among this set of solutions. This brings us to the Pareto terminology and the definitions adopted in multi-objective optimization.

Let us define a solution $x \in X$ with the objective function vector, $u = (u_1, u_2, ..., u_K)$ Since the formulations in thesis are all minimization problems, by definition of *Pareto dominance* if there no such solution $y \in X$ with the objective function vector, $v = (v_1, v_2, ..., v_K)$ satisfies that for $\forall k \in \{1, ..., K\}$, $u_k \geq v_k \bigcap \exists k \in \{1, ..., K\}, u_k > v_k$, then solution x is said to be a *Pareto optimal* solution. In other words, a solution is Pareto optimal if there exists no any other feasible solution which would improve some criterion without worsening at least one other criterion. Since the different objectives are not given a priori weights, all these *non-dominated* solutions form a set called *Pareto optimal set*. The aim in multi-objective optimization problems is to find all (or at least a representative set of) Pareto optimal set to from the true *Pareto front*.

Coello (2000), Konak *et al.* (2006), and Zitzler *et al.* (2004) provide comprehensive

surveys about the heuristic techniques for solving multi-objective problems. Evolutionary algorithms are even referred as "*the*" method for exploring the Pareto-optimal front in multi-objective optimization problems (Deb, 2001). The primary reason for using multi-objective evolutionary algorithms is their capability to find multiple Pareto optimal solutions in one single run (Jones *et al.*, 2002). As MOEA is a metaheuristic approach using evolutionary operators, their attempt is to find acceptable but approximate Pareto-optimal solutions.

A successful MOEA must be capable of obtaining an approximate Pareto front as close as possible to the true Pareto front (Konak *et al.*, 2006). Preferably, the members of the approximate Pareto front should be a subset of the Pareto optimal set. In addition, the members of the approximate Pareto front should be distributed homogeneously and should be diverse over of the in order to provide the decision-maker a comprehensive picture of about the trade-offs. In other words, the approximate Pareto front should represents the whole range of the true Pareto front.

Pierreval and Plaquin (1998) study a two-objective problem considering cell workload and traffic between cells as objectives. To solve the manufacturing cell formation problem the formulated, they apply a multi-objective evolutionary algorithm called niched Pareto evolutionary algorithm.

Coello and Christiansen (1998) propose two genetic algorithm based solution methods that forms a basis to many other algorithms. Non-dominated sorting genetic algorithm is first proposed by Srinivas and Deb (1994) and later on, modified and an improved version of it proposed by Deb *et al.* (2002). As for the applications of the multi-objective evolutionary algorithms, Coello and Lamont (2004) discuss the applications of multi-objective evolutionary algorithms with providing real world case studies.

In Zitzler *et al.* (2000), comparison is made between five alternatives of multi-objective evolutionary algorithms each representing different fitness assignment and

selection procedures with different ways to maintain the diversity concerns. Among these alternatives, non-dominated with its elitist structure and success in preventing premature convergence is concluded to be the most successful choice. As a consequence, the multi-objective genetic algorithm developed in this thesis mainly consults the operator schemes and methodology proposed by non-dominated sorting genetic algorithm.

As for the robustness concerns in multi-objective optimization problems and multi-objective evolutionary algorithms, Deb and Gupta (2005) describes the behavior of the Pareto front with respect to the changes in variables. Two types of robustness are defined for the solutions in the Pareto front and some conclusions are drawn whether the obtained frontier is robust to the perturbations in the variables. However, this study treats the robustness issue in a more like a sensitivity analysis concept. Li *et al.* (2005) proposes a multi-objective genetic algorithm which they called robust multi-objective genetic algorithm whose one objective function is identified as the robustness index. They treat robustness concerns as a separate objective and again based on the sensitivity estimations. Their study incorporates only two objective problems of which one is the single objective of the initial problem and no generalization is provided for higher dimensional problem cases.

In Iancu and Trichakis (2013), robust optimization problems are treated as multi-objective problems where each uncertainty scenario is considered as a separate objective. Their aim is to find robust Pareto optimal solutions instead of only robust solutions. This proposition is addressed in our generic algorithm and enables us to obtain robust optimal solutions existing in the Pareto front and compare the quality of our different formulations. They claim that the resulting problem has the same complexity with the underlying robust optimization problem and provide numerical studies related with the topics in portfolio optimization, inventory and project management.

3. PROPOSED MODELS

In this section, we present seven different mathematical programming formulations of QAP based facility layout problem which contain stochastic and/or robust constraints and/or objectives. As the definition of performing well varies from application to application, choosing an appropriate performance measure is part of the modeling process. These formulations are formed to enable us to have a broader view about the effects of different stochastic and robust optimization methodologies.

Since all of the formulations provided in this thesis use QAP as a basis framework, the same notation is used for all the formulations. The summary for definitions of sets, parameters and decision variable used in the formulations is as follows:

Table 3.1. Sets, parameters, and decision variable.

Sets:	
I	Set of departments
K	Set of locations
S	Set of scenarios
Parameters:	
f_{ij}^s	Flow between departments i and j under scenario s
d_{kl}	Distance between locations k and l
p_s	Probability that scenario s occurs
p	Desired robustness level, $p \geq 0$
Decision variable:	
x_{ik}	Binary variable which takes value of 1 if department i is located at location k, and 0 otherwise

The sets and the decision variable are common for all the formulations; whereas

not all of the parameters are used in each formulation. Some of these parameters are identified specifically for the requirements of some formulations.

Before presenting the seven formulations developed in this paper, it will be instructive to provide a brief explanation about the conventional QAP itself. The integer nonlinear programming formulation below forms the basis of the rest of the formulations developed and will be referred as "*classical QAP*" at the remaining part of this thesis:

Classical QAP formulation:

$$\min \sum_{i,j,k,l} f_{ij} d_{kl} x_{ik} x_{jl} \tag{3.1}$$

s.t.

$$\sum_{i} x_{ik} = 1 \qquad \forall k \tag{3.2}$$

$$\sum_{k} x_{ik} = 1 \qquad \forall i \tag{3.3}$$

$$x_{ik} \in \{0,1\} \qquad \forall i,k \tag{3.4}$$

Costs are assumed to occur from the movements between the departments and fixed cost of opening a department is not considered. The objective is to find an assignment of all departments to all locations such that the total travel distance of the assignments is minimized. The first constraint set ensures that every location is used by exactly one department and the second constraint ensures that every department gets exactly one location.

To solve the classical QAP, a genetic algorithm is developed referring to the procedures of past applications in literature. The steps and the performance of the developed algorithm will be explained in Chapter 4 in detail.

The classical QAP model is valid when both the flow and the distance matrices

are assumed to be known deterministically. However, in this thesis we try to develop ways of dealing with the uncertainty that might be present in these parameters. The formulations developed in this thesis use a set of scenarios to express the variability of the input external parameters. Each formulation suggests a different approach for getting protection from the uncertainty.

Before introducing the stochastic and robustness measures presented in this thesis, let us define S, the set of the scenarios. For each scenario s, where $s \in S$, there is a different deterministic minimization problem having a specific flow matrix. In this paper, only the flows between the departments are defined distinctively for each possible scenario. The product mixes and the quantities of each product type produced can easily change due to various reasons such as changes in demand or changes in the production preferences. On the other hand, all scenarios share the same distance matrix. In other words, distance matrix is not affected from the scenario realizations. It is not meaningful that the distances between candidate locations will change in different scenarios and the decision maker will be made to specify a different distance matrix for each scenario. Even so, this assumption can be easily modified and the solution procedures and results can be extended for the cases having also specific distance matrices for each scenario.

3.1. Minimizing Expected Cost

In the first model, the objective is to minimize the expected cost of the assignment of departments to locations under all possible scenarios in the scenario set. This model includes stochastic component in the objective function and constraints are the same as in the classical QAP.

Model 1:

$$\min \sum_{i,j,k,l,s} p_s f_{ij}^s d_{kl} x_{ik} x_{jl} \tag{3.5}$$

s.t.

$$(3.2), (3.3), and\ (3.4)$$

The parameter p_s, the probability that scenario s occurs, is used specifically in this formulation. Stochastic optimization requires some degree of probability information that is sometimes provided as probability mass/density functions or in our case distinct probability values assigned for each scenario. Obviously, as all probable scenarios are represented in the scenario set S, the sum of their probabilities must be equal to one.

$$\sum_s p_s = 1 \qquad\qquad s \in S \tag{3.6}$$

In this research, all p_s values are assumed to be equal without loss of generality. In other words, each scenario is equally likely to happen.

$$p_s = \frac{1}{|S|} \qquad\qquad s \in S \tag{3.7}$$

Solving this formulation is identical to solving the classical QAP. The only difference is that instead of a single set of flow parameters, we now have $|S|$ of them. By taking a weighted average with respect to the p_s values, we can fit all flow parameters into a single matrix and reduce the problem into the classical QAP.

$$f_{ij}^* = \sum_s p_s f_{ij}^s \qquad\qquad \forall i, j \tag{3.8}$$

The classical QAP created by replacing Equation 3.5 with 3.7 in the previous model.

$$\min \sum_{i,j,k,l} f^*_{ij} d_{kl} x_{ik} x_{jl} \tag{3.9}$$

s.t.

$$(3.2), (3.3), and\ (3.4)$$

3.2. Minimizing Maximum Cost

In the following three formulations, robust optimization methodology is practiced. As mentioned earlier, robust optimization approach often attempts to optimize the worst-case performance of the system. This minimax structure makes robust optimization problems more difficult to solve than stochastic optimization problems.

Two of the most common robutness measures that have been applied to facility layout problems in the literature are *minimax cost* (minimize the maximum cost across scenarios) and *minimax regret* (minimize maximum regret across scenarios). The primary attraction of minimax measures is that they do not require decision maker or planner to estimate scenario probabilities. They can be used when no probability information is known about uncertain parameters.

The *minimax cost solution* minimizes the maximum cost across all scenarios and thus gives emphasis to the worst case. The minimax cost formulation is as follows:

Model 2:

$$\min C_{max} \tag{3.10}$$

s.t.

$$\sum_{i,j,k,l} f_{ij}^{s} d_{kl} x_{ik} x_{jl} \leq C_{max} \qquad s \in S \tag{3.11}$$

$$(3.2), (3.3), and\ (3.4)$$

The variable C_{max} is defined as the maximum travel distance across all scenarios and the objective is to minimize this parameter. Solving this model is again similar to solve classical QAP when a genetic algorithm is employed. The only difference is for calculating fitness values of the generated solutions. To calculate fitness of a solution, total travel distance computation is made for each scenario and maximum of those values is taken as fitness of the solution.

3.3. Minimizing Maximum Regret

The term regret is usually used in robust optimization terminology. The regret is defined as the difference between the cost of a solution in a given scenario and the cost of the optimal solution for that scenario. The *minimax regret solution* minimizes the regret across all scenarios. The regret can be computed as either absolute or relative difference and both of these two measures can be transformed into each other.

There is no need to develop models with the objective of minimizing expected (average) regret. The reason is minimizing expected regret is equivalent to minimizing expected cost (Snyder, 2006).

3.3.1. Minimizing Maximum Absolute Regret

When the difference between the cost of a solution in a given scenario and the cost of the optimal solution for that scenario is defined in terms of nominal units, this is called the *absolute regret* of a solution. In our QAP formulation, absolute regret, R^{abs}, of a solution for each scenario is calculated as follows:

$$\sum_{i,j,k,l} f_{ij}^{s} d_{kl} x_{ik} x_{jl} - c_s^* = R^{abs} \qquad s \in S \tag{3.12}$$

where, the parameter c_s^* is the cost of the optimal solution for the scenario s and it is an input to the regret model. The c_s^* values are assumed to have been computed already by solving $|S|$ separate and deterministic QAPs. The variable R_{max}^{abs} is defined as the maximum regret across all scenarios and the objective is to minimize this parameter.

Thus, the third formulation which is called *absolute minimax regret solution* is as follows:

Model 3:

$$\min R_{max}^{abs} \tag{3.13}$$

s.t.

$$\sum_{i,j,k,l} f_{ij}^{s} d_{kl} x_{ik} x_{jl} - c_s^* \leq R_{max}^{abs} \qquad s \in S \tag{3.14}$$

$$(3.2), (3.3), and\ (3.4)$$

3.3.2. Minimizing Maximum Relative Regret

Minimizing relative regret formulation is very similar with the absolute regret formulation except the way how regret is defined. When the difference between the cost of a solution in a given scenario and the cost of the optimal solution for that scenario is defined in terms of the percentage value of the optimal solution for that

scenario, the *relative regret*, R^{rel}, of this solution is found. In our QAP formulation, relative regret of a solution for each scenario is calculated as follows:

$$\frac{\sum_{i,j,k,l} f_{ij}^s d_{kl} x_{ik} x_{jl} - c_s^*}{c_s^*} = R^{rel} \qquad s \in S \tag{3.15}$$

The *relative minimax regret solution* is as follows:

Model 4:

$$\min R_{max}^{rel} \tag{3.16}$$

s.t.

$$\frac{\sum_{i,j,k,l} f_{ij}^s d_{kl} x_{ik} x_{jl} - c_s^*}{c_s^*} \leq R_{max}^{rel} \qquad s \in S \tag{3.17}$$

$$(3.2), (3.3), and\ (3.4)$$

These four formulations given above use either stochastic optimization or robust optimization alone. Combining these two methodologies may create solutions that are satisfactory in terms of both methodologies. The subsequent formulation is developed to achieve this goal.

3.4. p-Robustness

In the fifth formulation, p-robustness concept is employed. The notion of p-robustness was first introduced by Kouvelis *et al.* (1992) in the context of facility layout without referring to this robustness measure as p-robustness. The term "*p-robust*" was first adopted by Snyder and Daskin (2006) to distinguish this robustness measure from other measures.

The aim of p-robustness approach is to combine the advantages of both the stochastic and robust optimization approaches. This is achieved by seeking the least-

cost solution in the expected value that is p-robust; i.e., whose relative regret in each scenario is no more than p, for given, $p \geq 0$. This robustness coefficient, p, defined as the maximum allowable regret and it is an external parameter that bounds the relative regret in each scenario.

In p-robustness, constraints are placed on the maximum regret rather than minimizing it. The relative regret is expressed in the same way as in the previous formulation. Thereby, p-robustness measure can combine minimizing expected cost by putting the stochastic measure into the objective function and putting the robustness measure in the constraints. The resulting robustness measure can be called as stochastic p-robustness.

In our formulations, the same regret limit is set for every scenario, even though it is sometimes convenient to specify a different regret limit p^s for scenarios. As in the case of Snyder and Daskin (2006), this parameter is fixed as $p = p^s$ for all $s \in S$. The mathematical model is as follows:

Model 5:

$$\min (3.5)$$

s.t.

$$\sum_{i,j,k,l} f_{ij}^s d_{kl} x_{ik} x_{jl} \leq (1+p) c_s^* \qquad s \in S \qquad (3.18)$$

$$(3.2), (3.3), and\ \ (3.4)$$

Obtained formulation is similar with the formulation that aims to minimize the expected cost except the fact that it also involves p-robustness condition. This additional constraint clearly narrows the feasible solution space and the effect of this tightening depends on the value that parameter p takes. Therefore, the choice of the parameter p is very important and an additional search is required for the decision maker. For instance, the constraint becomes redundant when $p = \infty$, and the formu-

lation becomes equal with the first formulation. On the other hand, for small values of p, there may be no feasible p-robust solutions for the problem. Thus, the p-robustness constraint creates a feasibility issue that is not present in the prior formulations.

3.5. Multi-objective Approaches

In the last two models, formulations with multi-objective functions are developed since one of the main goals is to produce solutions that perform well in many aspects and concerns. In the first multi-objective formulation, objective functions of the two formulations mentioned are combined aiming to develop a formulation that is able to meet the requirements of both formulations. In the second approach, the objective function is composed by a vector function whose elements represent the objectives of the solution in each scenario separately.

3.5.1. Minimizing Expected Cost and Absolute Regret

This formulation involves a multi-objective function aiming to obtain a set of feasible solutions that are satisfactory in terms of being robust to all possible scenarios. In this formulation, the solutions are represented by 2-dimensional vectors composed of expected cost and maximum absolute regret. A similar approach can be developed with relative regret, but we omit this here since its effect will be similar to the p-robustness case. Similarly, other robustness measures like minimizing maximum cost case can also be employed as the second objective.

Model 6:

$$\min_x \vec{F}(x) = \{f^1(x), f^2(x)\},$$

$$where\ \ f^1(x) = \sum_{i,j,k,l,s} p_s f^s_{ij} d_{kl} x_{ik} x_{jl}\ \ and\ \ f^2(x) = R^{abs}_{max} \tag{3.19}$$

s.t.

$$(3.2), (3.3), (3.4), and\ \ (3.14)$$

3.5.2. The Multi-objective QAP (mQAP)

In mQAP introduced by Knowles and Corne (2002), different flow types are considered as multiple objectives. In our case each scenario becomes a separate objective so number of objective functions is equal to the number of scenarios.

Model 7:

$$\min_x \vec{C}(x) = \{c^1(x), c^2(x), ..., c^{|S|}(x)\},$$

$$where\ \ c^s(x) = \sum_{i,j,k,l} f_{ij}^s d_{kl} x_{ik} x_{jl} \qquad s \in S \qquad (3.20)$$

s.t.

$$(3.2), (3.3), and\ \ (3.4)$$

The motivation behind developing the mQAP formulation is the fact that robust optimization problems can be seen as a multi-objective optimization problems with objectives corresponding to uncertainty scenarios. The optimum solutions for all stochastic and robustness measures covered in this thesis must fall within the Pareto front obtained from the mQAP formulation. This phenomenon called "*Pareto robust optimal solutions*" by Iancu and Trichakis (2013) briefly explains that for every robustness measure one of the robust optimal solutions should also be a Pareto optimal solution. This indicates that if there is only one robust optimal solution, then it must be a member of the Pareto front.

Mathematical propositions for specific robustness measure are presented in Aissi *et al.* (2009) stating that in a combinatorial minimization problem at least one optimal solution of minimax cost and minimax regret versions of the problem must be a Pareto optimal solution. It means that among all members of Pareto front, one solution must have a minimum maximum cost or minimum maximum regret value. The strong relationship between the robustness measures and the multi-objective version of our benchmark problem enables us to obtain optimal solutions in terms of all robustness

measures covered in this thesis if we find all members of the true Pareto front.

Similar proposition can also be offered for the expected cost measure. A solution that is optimal in terms of the expected cost criterion must also be a Pareto optimal solution for our mQAP formulation. The difference is that in the case of multiple optimal solutions for the expected cost criterion, not only at least one of the optimal solutions but all of them must also be Pareto optimal solutions.

Proposition 3.1. *The optimal solution for the expected cost criterion must be a non-dominated solution.*

Proof. Let us define $x \in X$ with the objective function vector, $u = (u_1, u_2, ..., u_{|S|})$ as a dominated solution and we want to show that it is an optimal solution for the expected cost criterion.

By the definition of Pareto dominance if x is a dominated solution, there must be at least one solution $y \in X$ with the objective function vector, $v = (v_1, v_2, ..., v_{|S|})$ satisfies that for $\forall s \in S, u_s \geq v_s \bigcap \exists s \in S, u_s > v_s$. However if this is the case then the expected cost of solution y is strictly less than the expected cost of solution x meaning than solution x cannot be an optimal solution for the expected cost criterion. Hence, by proof with contradiction, the optimal solution for the expected cost criterion must be a non-dominated solution. ■

By using mQAP model, instead of evaluating each performance measure separately, we can approximate the Pareto front in a single algorithm dedicated to mQAP formulation and select the appropriate member based on our chosen performance measure.

Our mQAP formulation can also include the p-robustness constraint by adding Equation (3.18) to the model and employ p-robustness condition after the formulation is solved and Pareto optimal solutions are obtained. Since both the objective space

and p-robustness constraints are defined for the same dimensions, each p-robustness constraint limits the feasible region by only one dimension and they are all orthogonal with one another. After the implementation of p-robustness condition, the decision maker will be able to reconsider his choice from the remaining Pareto optimal solutions, again depending on the robustness parameter p.

4. SOLUTION METHODS

This chapter explicates the solution technique used to solve the conventional integer linear formulation of QAP and other seven QAP-based stochastic and robust formulations developed in this thesis. The choice of the solution method takes into consideration of the features of the QAP itself as well as the formulations created. QAP is a combinatorial optimization problem in which either exact or heuristic methods can be used to solve. Exact algorithms include branch-and-bound (BB), dynamic programming (DP), cutting plane (CP) and branch-and-cut (BC) techniques.

As QAP is computationally NP-hard (Sahni and Gonzales, 1976), large problem instances can require a great amount of time to be solved optimally by exact methods. Thus, metaheuristic solution approaches like simulated annealing (SA), genetic algorithms (GA), scatter search, ant colony optimization (ACO), tabu search (TS) and variable neighborhood search (VNS) have become viable choices. On the other hand, the multi-objective structure of some of our formulations leads us to the use of a GA based algorithm since such algorithms are shown to work very satisfactorily in multi-objective domains. Therefore, to remain compatible and comparable, GA becomes natural choice to use across all formulations proposed.

4.1. Genetic Algorithm (GA)

4.1.1. Overview of the GA

This section explains the features of the employed GA and its variations regarding mainly the objective functions of formulations developed. Parameters and operator schemes are going to be presented as well as the pseudo codes for single objective and multi-objective genetic algorithms in the consecutive sections.

Genetic Algorithm (GA) concept was first introduced by Holland (1975) and be-

came one of the most popular heuristic algorithms to solve optimization problems. Basically, GAs imitates the process of evolution on solving an optimization problem by treating each feasible solution as an individual. The objective function values determines the fitness of the corresponding individuals. A population is formed by a subset of individuals.

GAs are general solution techniques, also known as *metaheuristics*, and they include a priori strategies like crossover and mutation that can be adapted to the specifications of problem structure. While implementing a GA, there is a substantial extent of flexibility in the choice and ordering of these strategies.

The general implementation of the GA includes operations of creating an initial population (preferably feasible). During their iterations, GAs contain one or more population of feasible solutions; such as parent population and child (offspring) population.

Certain types of operations can be executed on these populations based on how these populations are defined. Various selection and crossover operator schemes inspired by laws of nature are designed to generate fitter offspring and they must be adapted specifically for our formulations.

In addition, mutation/immigration and local search operators may also be added into GAs due to diversification and intensification concerns respectively. By striking a balance between intensification and diversification concerns, the developed GA will result in obtaining qualified and diversified solutions.

At the end of this section, the main loop of GA developed specifically for the single objective formulations including all mentioned operators will be presented and illuminated.

4.1.2. Solution Representation and Fitness Calculation

An individual is encoded by a chromosome composed of n genes each represents the location index that departments can be assigned to. This representation is schemed in Figure 4.1.

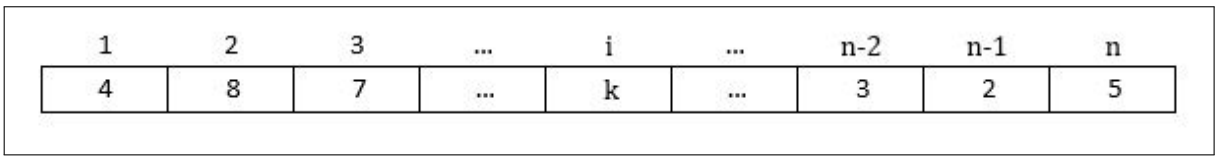

Figure 4.1. Solution Representation.

In Figure 4.1, department 1 is assigned to location 4, department 2 is assigned to location 8 and so forth. In general, values in this permutation store the assignment of department i to location k. Ensuring all values are different from each other, all possible permutations produce feasible solutions. Therefore, it can be concluded that the number of feasible solutions in a QAP of size n is $n!$.

Fitness value of each individual is simply determined by calculating the objective function values of the corresponding solution. So, for the first four models fitness value calculation is straightforward.

However, in p-robustness formulation, the infeasibility occurring due to the p-robustness constraint can be handled by adding a penalty function to the objective. Thus, fitness value indicates a penalty term resulting from the violation of the p-robustness constraints, in addition to the expected cost. The amount of penalty is not treated as a fixed term but it dynamically changes during the course of the search. Namely, as the number of iterations of GA increase, penalty of an infeasible solution will increase, so that the search will be directed towards feasibility. Furthermore, penalty of each individual will be calculated independently. Such an approach of penalizing is classified as "*variable penalty*" in Gen and Cheng (1996). Our proposed variable penalty calculation procedure works as follows: Expected cost of an individual is calculated only once at the creation. The fitness of a given individual in iteration

t is:

$$(fitness)^t = -((expectedcost) + p^t_{coef} * (violation)) \tag{4.1}$$

In Equation 4.1, the penalty coefficient, p^t_{coef}, is the penalizing multiplier to normalize a unit amount of violation in the objective function. In this thesis, the variability is caused by manipulating the penalty coefficient based on the iteration number, t, of the genetic algorithm.

The intuitive thought for this coefficient is to initiate it with small values and increase it exponentially to ensure that none of the solutions is infeasible at the last generation of the algorithm. The initial value of the penalty coefficient is chosen to be small compared with the expected cost component of fitness value to allow infeasible solutions at the earlier iterations of the algorithm.

$$p^{initial}_{coef} = p^0_{coef} = 0.01 * (expectedcost) \tag{4.2}$$

Then, the penalty coefficient is increased exponentially to ensure that at the end of the iterations, an individual violating the p-robustness constraints is not allowed to exist in the final population.

$$p^{final}_{coef} = p^{maxIter}_{coef} = 100 * (expectedcost) \tag{4.3}$$

The multiplier for the penalty coefficient depends on the predetermined initial and final values of the penalty coefficient and also the number of total iterations in the

algorithm.

$$(multiplier)^{maxIter} = \frac{p_{coef}^{final}}{p_{coef}^{initial}} \tag{4.4}$$

In each iteration, penalty coefficient and therefore fitness values of all members of the population are updated by using the following rule.

$$p_{coef}^{t+1} = (multiplier) * p_{coef}^{t} \tag{4.5}$$

It must be noted that this update mechanism is unique for the p-robustness formulation and none of the other formulations has a dynamic fitness value calculation procedure.

4.1.3. Initial Population Generation

The creation of the initial population certainly affects the final population of a GA. The performance measures for an initial population are the average fitness value and the diversity within the population. There are several constructive heuristics suggested in QAP literature, such as GRASP (Pardalos and Resende, 1994) aiming to perform well under these criteria. However, the structure of this methodology does not fit to our case due to the presence of scenarios since it is not possible to calculate the move cost at each greedy step when there are more than one scenario.

One possible alternative is to produce individuals with random permutations. The numbers from 1 to n are mixed in a random manner so that the result reflects a distinct individual. Generating individuals in this way creates no problem about infeasibility issue and requires minor computational burden. Diversity concerns will also be satisfied as all individuals are spread out through the solution space equally. For this reason, when generating the initial population, individuals are produced in a

random manner.

The only exceptional individuals are the optimal solutions achieved from solving $|S|$ separate scenario QAPs. Our preliminary analysis supports the idea that including these individuals in the initial population increases the quality of our GA. Briefly, after adding the optimal solutions of scenario QAPs,the rest of the initial population is generated in a random manner and the algorithm for generating the initial population is provided in Figure 4.2.

Algorithm **"*Random*" Generate Initial Population**
Initialize *populationSize*
Initialize *population* : $P \leftarrow \emptyset$
Add scenario optimals to P
Initialize $i \leftarrow |S|$
while $i <$ *populationSize* **do:**
    Generate a solution with random permution and add it to P
    $i \leftarrow i + 1$
end while

Figure 4.2. Pseudo Code for "*Random*" generating initial population.

4.1.4. Selection and Crossover

The performance of GA is often very sensitive to parent selection strategy and crossover operators. Selection procedure refers to the decision process of parents that will apply the crossover operator. Three alternative selection procedures and two alternative crossover operators are built and experimented during our research after a detailed investigation of the related literature. The selection procedures are named as "*random*", "*binary tournament*" and "*modified binary tournament*", and the crossover operators are named as "*one-order*" and "*path swap*". In the literature, selection criteria typically give a higher priority to individuals with better fitness values as this is

the case in the last two of the selection procedures adopted and experimented in this thesis. In contrast, each individual is given equally likely chance to be selected in the first selection procedure.

Algorithm **"*Random*" Selection**
Initialize $parentSize$, $population : P$
Initialize $parentPopulation : P_{par} \leftarrow \emptyset$
Initialize $i \leftarrow 1$
while $i < parentSize$ **do:**
    Add a random individual from P to P_{par}
    $i \leftarrow i + 1$
end while

Figure 4.3. Pseudo Code for "*Random*" selection.

In binary tournament selection procedure, individual with a better fitness value is favored among the two individuals that have been chosen from the population P. In case of tie between the candidates, one of them is selected randomly. The comparison between the candidate individuals is made with respect to their fitness values.

Algorithm **"*Binary Tournament*" Selection**
Initialize $parentSize$, $population : P$
Initialize $parentPopulation : P_{par} \leftarrow \emptyset$
Initialize $i \leftarrow 1$
while $i < parentSize$ **do:**
    Choose two random individuals from P as candidates
    Compare candidates and add fitter candidate to P_{par}
    $i \leftarrow i + 1$
end while

Figure 4.4. Pseudo Code for "*Binary Tournament*" selection.

To take into account the genotypes, we have proposed a selection procure called modified binary tournament. In modified binary tournament, first parent is selected in the same way with binary tournament only considering fitness values of two candidate individuals. In contrast, second parent is selected considering the similarity between the candidates and already chosen parent. Similarity between two individuals is defined as the number of genes they have in common. The candidate individual which has less number of common genes with the first parent is selected as the second parent. In this way, parents with different characteristics go into crossover operation in order to diversify.

Algorithm "***Modified Binary Tournament***" **Selection**
Initialize $parentSize$, $population : P$
Initialize $parentPopulation : P_{par} \leftarrow \emptyset$
Initialize $i \leftarrow 1$
while $i < parentSize$ **do:**
    Choose two random individuals from P as candidates
    Compare candidates and select fitter candidate as first parent
    Choose two random individuals from P as candidates
    Evaluate similarity between candidates and first parent
    Select the dissimilar candidate as second candidate
    Add first and second parents to P_{par}
    $i \leftarrow i + 2$
end while

Figure 4.5. Pseudo Code for "*Modified Binary Tournament*" selection.

As for the crossover operator schemes, two different approaches are tested. First of these approaches, called order crossover (Davis, 1985), is more conventional and has applications in different types of encoding schemes than the permutation representation employed in this thesis. In our research we used one-order crossover operator scheme.

In one-order crossover, two parents generate two children by cutting the chromosomes of both parents into two pieces at a random point and switching the second parts of their chromosomes. The infeasibilities caused from duplicated and unapparent genes are repaired by following the order of permutations in the parents.

In path crossover approach proposed by Ahuja *et al.* (2000), the aim is again producing offspring that combines good characteristics of both parents. The "*path*" connecting the parents is formed by creating new solutions through move operations (i.e. swap, insert) that make the parents look more alike at each step. It is stated that swap operator performs better.

In path swap operator scheme, starting from a random position of the chromosomes, the alleles of the two parents at that position are examined. If the corresponding alleles of two parents are different, a swap operation is performed within both parents. Two resulting solutions are inspected and whichever solution is fitter, a move is made by the corresponding parent, thus forming nodes on the path. In the next iteration, this new node is considered as the parent and compared with the other parent. In each swap operation, two parents look more alike and the path swap scheme continues until all alleles of two parents become the same and the full path is formed.

Path swap crossover is illustrated in Figure 4.6. In this iteration of swap operation, first position is examined. In the first parent, department 5 is at first location and in the second parent department 2 is at first location. There are two choices that two parents can move closer to each other: by swapping the locations of the departments 5 and 2 in the first parent or by swapping the locations of the departments 2 and 1 in the second parent. After performing two swap operations, fitness of the resulting two individuals are compared and the swap operation is performed for which the corresponding individual has a better fitness.

Once the path is built, the nodes are considered as candidate children. The fittest member of the path is selected as the first child. In addition to that, one of the initial

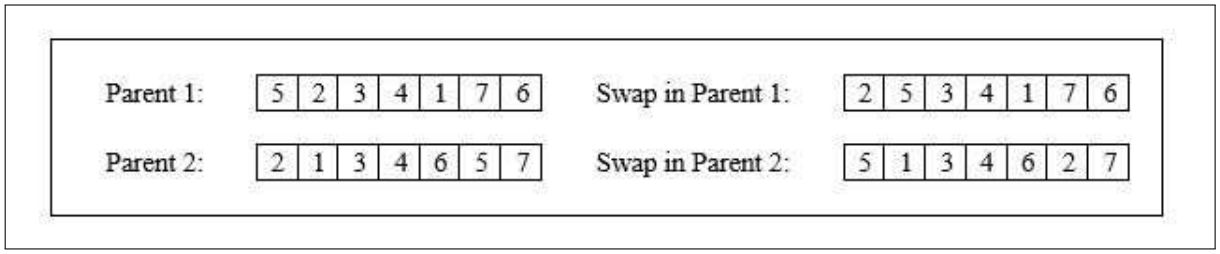

Figure 4.6. Illustration for path swap crossover.

parents is chosen as the second child based on fitness value comparison between two parents. If the generated child is fitter than both parents, the parent which is less similar to the first child (having less number of alleles in common) is chosen as second child. Pseudo code for path swap crossover is presented in Figure 4.7.

Algorithm **"*Path Swap*" Crossover**

Initialize $parentPopulation : P_{par}$

Initialize $childPopulation : P_{chi} \leftarrow \emptyset$

Initialize $i \leftarrow 1$

while $i < parentSize$ **do:**

    Initialize $parent1 \leftarrow P_{par}[i]$, $parent2 \leftarrow P_{par}[i+1]$

    Initialize $child1 \leftarrow \emptyset$, $child2 \leftarrow \emptyset$

    List all different indices between $parent1$ and $parent2$

    Initialize $pathPopulation : P_{path} \leftarrow \emptyset$

    foreach *index* **in** list of different indices

        Perform swap to both parents

        Among transfered parents, add the one with better fitness to P_{path}

    end foreach

    Select the fittest child in P_{path} as $child1$

    if $child1$ is fitter than both parents

        Select parent that is dissimilar with $child1$ as $child2$

    else

        Select fitter parent as $child2$

    end if

    Add $child1$ and $child2$ to P_{chi}

    $i \leftarrow i + 2$

end while

Figure 4.7. Pseudo Code for "*Path Swap*" crossover.

4.1.5. Immigration and Local Improvement

Immigration operator, a variation of mutation operator, aims to increase diversity in the population by introducing new individuals that significantly differ from the individuals so far. In each iteration, some pre-specified percentage of the current

population is removed and newly introduced solutions, called *immigrants*, replace these members.

The individuals that will be replaced by the immigrants are determined after all members of the transient population have been sorted based on the fitness values. Once the poorest members of the population are identified, they are substituted with the immigrants created.

For the generation stage of immigrants, Ahuja *et al.* (2000) propose a process that makes use of the historical frequency information similar to the long-term memory in TS literature (Glover and Laguna, 1999). In this approach, the aim is to produce immigrants that have genetic representation significantly different from the individuals generated until that moment.All past and present information regarding the number of times each department has been assigned to each location is stored in a nxn matrix called "*population history*". The higher values in the population history matrix indicate that the corresponding assignment is observed in many individuals whereas the lower values indicate fewer individuals in the past and present populations have the corresponding assignment in their permutation. The methodology they used targets to create new assignments including the lower values in the population history matrix by choosing departments in a random sequence and disperse them to the lowest possible location assignment. This procedure allows to search unexplored regions of the solution space.

However, by generating immigrants with this methodology, there is a high chance that the fitness of immigrants will be below than other members already existing in the current population. The outcome is that they rarely qualify as parent individuals and contribute to create new offspring. In addition, it is likely for them to be eliminated at the very next iterations. Therefore, all immigrant individuals are subjected to the local improvement operation right after their creation. In this way, the unexplored regions of the solution space are intensively investigated and individuals with hopefully fair fitness values are incorporated into the population which will lead to improve the

overall quality. Pseudo code for immigration is presented in Figure 4.8.

Algorithm **Immigration**
Initialize $migrateCount \leftarrow immigrationRate * populationSize$
Initialize $population : P$, $immigrantPopulation : P_{mig} \leftarrow \emptyset$
Initialize $i \leftarrow 1$
while $i < migrateCount$ **do:**
    Generate new individual called *immigrant* based on $populationHistory$
    Compare candidates and select fitter candidate as first parent
    Apply **local improvement** to *immigrant*
    Add *immigrant* to P_{mig}
    $i \leftarrow i + 1$
end while
Replace worst $migrateCount$ members of P with P_{mig}

Figure 4.8. Pseudo Code for immigration.

As for the local improvement operator, the 2-exchange neighborhood local search is used. In 2-opt local search, neighbor solutions are generated by swapping two genes in the permutation of an existing solution. If fitness value of a candidate neighbor individual is better than fitness value of an existing individual, the existing individual is transformed into this candidate neighbor and the same local search procedure is restarted. This local improvement process is carried on until an individual having a fitness value better than whole of its neighborhood is obtained. Thus, it is guaranteed that the consequential individual is at a 2-opt local optimum. Pseudo code for local improvement is presented in Figure 4.9.

Some pre-specified percentage of the current population are placed into local improvement operation in each iteration. This percentage is determined by the local improvement rate parameter. However, as the individuals obtained from immigration have already entered the local improvement, the effective local improvement rate is the

higher the actual local improvement rate.

```
Algorithm "2-Opt" Local Improvement
Initialize localImprovementRate, populationSize
Initialize population : P, currentSolution ← ∅, neighborSolution ← ∅
Initialize i ← 1
while i < populationSize do:
    if rand(0, 1) < localImprovementRate
        currentSolution ← population[i]
        Initialize flag ← true
        while flag do:
            flag ← false
            foreach neighborSolution in neighbor(currentSolution)
                Compare currentSolution and neighborSolution
                if neighborSolution is better than currentSolution
                    currentSolution ← neighborSolution
                    flag ← true
                end if
            end foreach
        end while
    end if
    i ← i + 1
end while
```

Figure 4.9. Pseudo Code for "*2-opt*" local improvement.

By adjusting the values of immigration rate and local improvement rate parameters properly, the one can ensure the genetic algorithm developed satisfies diversification and intensification concerns. These two parameters can have values between zero and one. Assigning values close to zero indicates the underuse and values close to one indicates the overuse of the corresponding operators. Thus, the choice for these parameters

must be done carefully. At the fine tuning stage of the genetic algorithm, several levels for both the immigration rate and local improvement rate are experimented.

4.1.6. The Main Loop of GA

The outline of the GA dedicated for single objective formulations is presented in Figure 4.10. After generating an initial population, the main loop of the algorithm which includes selection, crossover, immigration and local improvement; runs for a certain number of iterations (or generations). In each iteration, after child population is generated, it is sorted based on fitness values and statistics regarding best solution and position history are updated. The same procedure is applied also after immigration and local improvement operators are executed.

Algorithm **Main Loop of GA**

Initialize $immigrationRate$, $localImprovementRate$, $populationSize$, $maxIteration$

Initialize $positionHistory$

Initialize $iterationCount \leftarrow 1$

Generate initial $population : P$

Update $bestSolution$ and $positionHistory$

Initialize $parentPopulation : P_{par} \leftarrow \emptyset$,

while $iterationCount < maxIteration$ **do:**

    Initialize $parentPopulation : P_{par} \leftarrow \emptyset$, $childPopulation : P_{chi} \leftarrow \emptyset$

    Select P_{par} from $population$

    Generate P_{chi} from P_{par} with **Crossover**

    Sort P_{chi}

    Update $bestSolution$ and $positionHistory$

    Apply **Immigration** to P_{chi}

    Apply **Local Improvement** to P_{chi}

    $P \leftarrow P_{chi}$

    Update $bestSolution$ and $positionHistory$

    $iterationCount \leftarrow iterationCount + 1$

end while

Figure 4.10. Pseudo Code for the main loop of GA.

4.2. Proposed Multi-objective Genetic Algorithm

In this section, we will introduce the proposed multi-objective genetic algorithm to approximate an efficient Pareto front in reasonable computation times.

The MOEA techniques are classified into different categories relying on the sequence of two main stages of the algorithm (Deb and Gupta, 2005). These two stages are searching the solution space with respect to objective functions and deciding on

what kind of tradeoffs between the priorities of the objectives are convenient from the decision maker perspective. Some techniques make decision about the priority ranks of the objectives before searching the solution space, whereas some do the reverse.

The choice in our research is to find as many as solutions in the Pareto front and then leave the rest of the multi-criteria decision making process to a decision maker. This is called "*A Posteriori*" technique. Among the sub-techniques in "A Posteriori" techniques, such as independent sampling, aggregation selection, and Pareto selection; an algorithm in the last sub-technique is selected. In Zitzler *et al.* (2000), some of the most applied MOEAs from different sub-techniques are compared and the performances are evaluated regarding with the success of having close approximation to the true Pareto front and diversity concerns not only about avoiding local optimums but also obtaining the entire Pareto front. Based on their findings, NSGA-II which has been developed by Deb *et al.* (2002) is adopted for solving our multi-objective formulations.

In NSGA-II, sorting of individuals requires the calculation of two attributes for each individual: *non-domination rank* and *crowding distance*. In the calculation stage of non-domination rank, all solutions are examined in regard to whether they are dominated by any other solution or not. Then, all non-dominated solutions are ranked as one. After excluding these non-dominated solutions, the same is done for the solutions in the remaining set and the non-dominated solutions are ranked as two and this procedure is implemented until all individuals in the entire population are ranked.

After the assignment of non-domination ranks, a front for each rank is formed and crowding distances are assigned. The goal in crowding distance assignment is to ensure that every region of a front is represented by enough number of individuals (i.e. the individuals are not crowded into certain regions) In the crowding distance computation, all solutions are assigned distance values equal to the absolute normalized difference in the objective function values of two adjacent solutions in the same front. This procedure is done for every objective function and distances are added up for each individual. At a result, individuals take place in the scarce regions of their fronts will

have high crowding distance values and they will be preferred for reproduction in GA.

After the assignment of two attributes, solutions are sorted primarily based on their non-domination ranks and solutions in the same front are sorted based on their crowding distance assignment.

In NSGA-II, the population and the child population generated are combined and sorted based on the non-dominated sorting criteria in each iteration. Since all previous and current members are included in the combined population, elitism is ensured. Then, the set of best individuals with size equal to population size parameter is kept and declared as the new generation of population.

Evidently, some modifications will be necessary on NSGA-II to address the specific characteristics of our problem. When generating the initial population, optimal solutions achieved from solving $|S|$ separate scenario QAPs are included in the initial population. We observed that this greatly enhances the capability of the search in representing the whole range of the Pareto front, by introducing genetic material from the extreme ends of the objective function space. Another enhancement that provided the search with valuable genetic information is achieved by making us of the final populations of the $|S|$ separate scenario QAPs. All final populations are combined and members of the first front of this combined population are added to the initial population. The remaining members of the initial population are generated randomly.

Preliminary results of our algorithm hinted the need for additional diversity. It is observed that the crowding distance assignment is inadequate to prevent premature convergence. An attribute called *duplication factor* is introduced to each individual. In duplication factor assignment, every unique solution is ranked as one and their replicas are ranked with the number of times the same solution is observed in the current population. Thus, it is preferred to have small values for the duplication factor and in our modification, solutions are sorted primarily based their duplication factor values and then sorted on the attributes proposed by NSGA-II. The outline of the

multi-objective GA is presented in Figure 4.11.

Immigration is implemented exactly in the same way as in our single objective GA implementation. However, in path crossover, binary tournament and local improvement procedures comparison of individuals requires some adaptations. In path crossover, after the swap moves the child that dominates the other is selected to be included on the path. After the path is complete, candidate children created on the path are sorted based on non-dominated sorting criteria. Similarly in binary tournament, non-dominated sorting criteria are used for parent selection. In local improvement, we move to a neighbor solution only if it dominates the current solution.

Algorithm **Main Loop of Multi-objective GA**

Initialize $immigrationRate$, $localImprovementRate$, $populationSize$, $maxIteration$

Initialize $positionHistory$

Initialize $iterationCount \leftarrow 1$

Generate initial $population : P$

Update $bestSolution$ and $positionHistory$

Initialize $parentPopulation : P_{par} \leftarrow \emptyset$,

while $iterationCount < maxIteration$ **do:**

    Initialize $parentPopulation : P_{par} \leftarrow \emptyset$, $childPopulation : P_{chi} \leftarrow \emptyset$

    Select P_{par} from $population$

    Generate P_{chi} from P_{par} with **Crossover**

    Apply *non-dominated sorting* on P_{chi}

    Update $bestSolution$ and $positionHistory$

    Apply **Immigration** to P_{chi}

    Apply **Local Improvement** to P_{chi}

    Update $bestSolution$ and $positionHistory$

    Combine P and P_{chi} as $combinedPopulation : P_{com}$

    Apply *non-dominated sorting* on P_{com}

    $P \leftarrow$ best $populationSize$ solutions in P_{com}

    Update $bestSolution$ and $positionHistory$

    $iterationCount \leftarrow iterationCount + 1$

end while

Figure 4.11. Pseudo Code for the main loop of multi-objective GA.

5. NUMERICAL RESULTS AND PERFORMANCE OF THE PROPOSED SOLUTION METHODS

In this chapter, performances of the solution methods developed in this thesis are presented with extensive numerical analyses. In Section 5.1, required experimental setting, fine tuning and the validation for our GA are displayed. Section 5.2 clarifies the methodology followed to generate the problem instances used. Finally, in Section 5.3, several types of numerical analysis regarding the evaluation and comparison of the performances of the different formulations are presented.

Our algorithms are coded in C# programming language in Microsoft Visual Studio 2013. All experiments are carried out on a PC with 3.60 GHz Intel®Core™ i7-3820 CPU and 32 GB RAM, running under 64-bit Windows 7 operating system.

5.1. Settings for the GA

In this section, experiments to investigate the effects of different types of operator schemes and different levels of parameters are designed. Operator schemes setting are presented in Section 5.1.1 and fine tuning of the parameter values are presented in Section 5.1.2. After completing these experiments, performance of our genetic algorithm will be verified with test problems and will be presented in Section 5.1.3.

5.1.1. Operator Schemes Setting

To build the most successful genetic algorithm, we should first to choose which operator schemes must be selected among the possible operator schemes created. There are three selection operator schemes and two for crossover operator schemes. The six different combinations tested are listed in Table 5.1.

Table 5.1. Combinations for testing operator schemes.

Abbreviation	**Selection**	**Crossover**
R- OO	Random	One-order
R - PS	Random	Path Swap
BT - OO	Binary Tournament	One-order
BT - PS	Binary Tournament	Path Swap
MBT - OO	Modified Binary Tournament	One-order
MBT - PS	Modified Binary Tournament	Path Swap

To make a comparison and choice between these alternatives ten problem instances with different sizes and properties from the QAPLIB (Burkard *et al.*, 1997) which is composed of a much-studied suite of problems and being referenced by numerous researches, are selected and 10 replications are made for each instance.

The comparison criteria among the operator scheme alternatives are the number of times the optimal solution is found in Table 5.2 and the solution time required for the algorithms to terminate in Table 5.3. The best results in each problem instance are bolded.

Form the results, it can be claimed that path swap crossover performs better than the one-order crossover since the settings using path swap crossover reach a higher number for finding the optimal solutions. Especially, the one with binary tournament selection scheme can be recognized as the most successful setting.

After investigating the time requirements, binary tournament scheme is observed to have the least time consuming among the selection operator alternatives as the minimum values always occur under that scheme. In addition, it can be noted that path swap crossover consumes less time than one-order crossover. As a result, binary tournament selection and path swap crossover operator setting is chosen for our GA.

Table 5.2. Number of optimal solutions found in 10 replications for each operator scheme setting.

Problem Name	**Problem Size**	**R-OO**	**R-PS**	**BT-OO**	**BT-PS**	**MBT-OO**	**MBT-PS**
chr12a	12	**10**	**10**	**10**	**10**	**10**	**10**
tai12a	12	**10**	**10**	**10**	**10**	**10**	**10**
esc16a	16	**10**	**10**	**10**	**10**	**10**	**10**
els19	19	4	8	3	8	6	**10**
had20	20	**10**	**10**	**10**	**10**	**10**	**10**
rou20	20	4	4	3	**5**	2	1
scr20	20	9	8	**10**	**10**	**10**	**10**
bur26a	26	**10**	**10**	**10**	**10**	**10**	**10**
kra30a	30	**6**	5	5	**6**	2	**6**
kra32	32	8	**9**	7	**9**	8	8
TOTAL		81	84	78	**88**	78	85

Table 5.3. Time spent (in seconds) for each operator scheme setting cumulative of 10 replications.

Problem Name	**Problem Size**	**R-OO**	**R-PS**	**BT-OO**	**BT-PS**	**MBT-OO**	**MBT-PS**
chr12a	12	9	8	9	**8**	9	9
tai12a	12	7	6	7	**6**	7	7
esc16a	16	10	11	**10**	11	10	11
els19	19	98	91	**88**	90	98	91
had20	20	94	83	85	**82**	95	83
rou20	20	39	35	38	**34**	39	35
scr20	20	45	40	44	**38**	44	41
bur26a	26	218	198	216	**194**	218	194
kra30a	30	319	270	312	**264**	321	271
kra32	32	396	347	389	**345**	394	349

5.1.2. Fine Tuning of Parameter Values

There are four main parameters in our genetic algorithm. These are the population size, number of maximum iteration, immigration rate and local improvement rate. The values of the first two parameters are determined after some pilot runs and both *populationSize* and *maxIteration* parameter values are determined as 100.

For *immigrationRate* and *localImprovementRate*, some levels are selected and full design of experiment is made using the preselected operator schemes. Same comparison criteria are used with Section 5.1.1 and the results are displayed in Table 5.4 and 5.5.

From the results, it can be noted that using immigration and local improvement excessively does not help algorithm to improve but increases the time requirement. As the local improvement rate increases, our algorithm faces the problem of premature convergence. Although there is no significant difference between the results of different levels, the decision for these two parameters are made in favor of the setting that reaches the highest number of finding the optimal solution. Therefore, after the fine tuning of the parameter values, *immigrationRate* is set to 0.4 and *localImprovementRate* is set to 0.2.

5.1.3. Validation of the GA

As the final choices are made for both the operator schemes and genetic algorithm parameters, the quality of the obtained solutions and the speed of solving the problems instances evaluated by solving test cases available in QAPLIB. 20 test problems covering a wide range of problem sizes and characteristics are selected to validate the performance of our GA to solve classical QAP formulation.

The primary evaluation criterion for our algorithm is the objective function values of the best solutions found. In order to improve the solution quality and reduce the

Table 5.4. Number of optimal solutions found in each parameter level.

Immigration Rate	**Local Improvement Rate**	**chr12a**	**tai12a**	**esc16a**	**els19**	**had20**	**rou20**	**scr20**	**bur26a**	**kra30a**	**kra32**	**TOTAL**
0,2	0,2	**10**	**10**	**10**	7	**10**	**5**	9	**10**	**6**	7	84
	0,4	**10**	**10**	**10**	**9**	**10**	4	9	**10**	4	8	84
	0,6	**10**	**10**	**10**	7	**10**	**5**	**10**	**10**	3	8	83
	0,8	**10**	**10**	**10**	**9**	**10**	**5**	**10**	**10**	3	6	83
0,4	0,2	**10**	**10**	**10**	8	**10**	**5**	**10**	**10**	**6**	**9**	**88**
	0,4	**10**	**10**	**10**	**9**	**10**	**5**	**10**	**10**	**6**	7	87
	0,6	**10**	**10**	**10**	7	**10**	**5**	**10**	**10**	3	8	83
	0,8	**10**	**10**	**10**	8	**10**	4	**10**	**10**	**6**	6	84
0,6	0,2	**10**	**10**	**10**	7	**10**	4	**10**	**10**	4	7	82
	0,4	**10**	**10**	**10**	7	**10**	2	**10**	**10**	4	8	81
	0,6	**10**	**10**	**10**	8	**10**	4	**10**	**10**	3	6	81
	0,8	**10**	**10**	**10**	6	**10**	4	**10**	**10**	5	8	83
0,8	0,2	**10**	**10**	**10**	7	**10**	**5**	9	**10**	**6**	7	84
	0,4	**10**	**10**	**10**	7	**10**	3	9	**10**	4	8	81
	0,6	**10**	**10**	**10**	6	**10**	3	**10**	**10**	**6**	8	83
	0,8	**10**	**10**	**10**	5	**10**	3	**10**	**10**	**6**	7	81

Table 5.5. Time spent (in seconds) for each parameter level cumulative of 10 replications.

Immigration Rate	**Local Improvement Rate**	**chr12a**	**tai12a**	**esc16a**	**els19**	**had20**	**rou20**	**scr20**	**bur26a**	**kra30a**	**kra32**
0,2	0,2	**7**	**5**	**10**	98	86	37	40	209	**226**	**313**
	0,4	10	8	12	123	98	**33**	48	225	342	380
	0,6	11	11	16	111	128	43	52	223	401	508
	0,8	12	8	18	127	135	56	66	280	518	543
0,4	0,2	8	6	11	**90**	**82**	34	**38**	**194**	264	345
	0,4	9	9	12	97	112	41	49	237	303	410
	0,6	13	10	18	152	97	43	48	273	375	410
	0,8	10	10	15	138	132	47	63	301	454	521
0,6	0,2	12	9	16	105	122	44	54	246	401	504
	0,4	12	9	17	148	136	51	59	287	427	500
	0,6	17	13	19	183	134	72	70	382	539	600
	0,8	16	14	20	203	188	76	74	401	547	742
0,8	0,2	15	10	21	212	180	70	79	450	572	607
	0,4	19	16	22	253	218	80	84	415	700	856
	0,6	18	15	36	256	284	108	106	525	644	1006
	0,8	25	21	36	257	243	89	123	757	937	1097

effect of the randomness, the solutions are replicated with 10 different seeds. Thus, both best and average objective function values obtained through 10 replications are presented in Table 5.6. For the last four problems the comparison is made with the best upper bound values available in the QAPLIB.

The columns in the right side presents the best value in the objective values found in 10 replications. The percentage gap between the optimal solutions provided and the best solution found by our GA is equal to zero in almost all of the problems. Only for the problems with size larger than 36, the gap between the best solution found by our algorithm and the optimal takes values greater than zero. Even so, the gap takes values about only one percent.

The columns in the left side present the average of objective values found in 10 replications. As for the average performance of our algorithm, it can be observed that our algorithm finds the optimal solution in every run for the smaller problems. In addition, the gap never reaches to three percent even for the larger problem sizes.

The time required in seconds for our algorithm to complete a single replication and 10 replications are also provided in Table 5.6. It takes only about 10 seconds to complete a replication to solve a problem of size 12, 5 minutes to solve a problem of size 30, and half an hour to solve a problem of size 150.

Taking all these results into consideration, it can be claimed that of our GA is adequate to solve classical QAP formulation since all test problems are solved successfully and optimal solutions are obtained in almost every test problem.

Table 5.6. Validation of proposed GA (CPU times are cumulative of 10 replications).

Problem Name	**Problem Size**	**Optimal Value**	**Best Objective**	**Best % Gap**	**Total Time (sec)**	**Average Objective**	**Average % Gap**
chr12a	12	9552	9552	0.0	84	9552	0.0
tai12a	12	224416	224416	0.0	65	224416	0.0
esc16a	16	68	68	0.0	108	68	0.0
els19	19	17212548	17212548	0.0	909	17301154	0.5
had20	20	6922	6922	0.0	826	6922	0.0
rou20	20	725522	725522	0.0	352	725759	0.0
scr20	20	110030	110030	0.0	403	110030	0.0
chr25a	25	3796	3796	0.0	1199	3856	1.6
bur26a	26	5426670	5426670	0.0	1979	5426670	0.0
kra30a	30	88900	88900	0.0	2695	89320	0.5
nug30	30	6124	6124	0.0	3866	6127	0.1
tai30b	30	637117113	637117113	0.0	5245	637371231	0.0
tho30	30	149936	149936	0.0	3933	150144	0.1
esc32a	32	130	130	0.0	2012	133	2.3
kra32	32	88700	88700	0.0	3469	88770	0.1
ste36a	36	9526	9548	0.2	8507	9633	1.1
tai40a	40	3139370	3178108	1.2	4237	3200030	1.9
sko72	72	66256	66346	0.1	10283	67938	2.5
wil100	100	273038	273786	0.3	15820	277172	1.5
tho150	150	8133398	8260482	1.6	19053	8322744	2.3

5.2. Creating Problem Instances

For numerical studies, QAP instances with different scenarios are needed. Scenarios corresponding to a problem instance will be defined by different flow matrices over a given distance matrix. Unfortunately, the problems given in the QAPLIB are not suitable for this purpose, therefore we generate our test problems randomly by means of a C# code.

To create the single distance matrix for each problem instance certain rules are followed. Each entity that represents the distance between the two locations is generated using a discrete uniform distribution with predefined upper and lower bound. In all problem instances, distance matrices are defined as symmetrical meaning that the distance from location k to l is the same with the distance from l to k. Finally, the diagonal entities of the distance matrix are always equal to zero.

For generating $|S|$ number of the flow matrices for each problem instance, a more complex arrangement is implemented. Flow matrices are not generated only from a random variable distribution. Each flow entity is assumed to be composed from several product flows. Prior to the generation stage of the flow matrices, a product pool is formed including products with distinct characteristics presented in Table 5.7. Each product visits the departments in a certain sequence and it has lower and upper bound values for its flow quantities. For each scenario, different combinations of products are selected randomly to form the aggregate flow matrix. The flow quantities are also generated randomly within the lower and upper bounds specified for each product meaning that even the same product is used in different flow matrices, its flow quantities vary from one another in each different realization of the product.

Table 5.7. Product features.

Product Name	**Department Count**	**Demand**	
		Lower Bound	**Upper Bound**
Product A	2	50	100
Product B	5	50	100
Product C	8	50	100
Product D	10	50	100
Product E	2	70	100
Product F	5	70	100
Product G	8	70	100
Product H	10	70	100
Product I	2	50	200
Product J	5	50	200
Product K	8	50	200
Product L	10	50	200
Product M	2	70	200
Product N	5	70	200
Product O	8	70	200
Product P	10	70	200

The difference between the flow matrices results not only from the random structure of product flows. In addition to that, the choice of which products are going to be used in also differs in each flow matrix. These two factors are combined to create the varying structure of the flow matrices. Similar with the procedure of generating distance matrix, flow matrices are assumed to be symmetric meaning that the flow from department i to j is the same with the flow from j to i. Last of all, the diagonal entities of the flow matrix are always zero too.

After generating a set of flow matrices and their corresponding distance matrix, the next task is to create problem instances. In our problem instance settings, each problem consists of *three* scenarios. Three scenarios allow 3-dimensional graphical analysis therefore it is helpful in terms of demonstration. Additionally, three scenarios

are also adequate to represent most real life applications. For instance; first scenario symbolizes high demand case, second is medium and the third is low demand case.

Scenario selection for problem instances is based on the best objective function values found in scenario QAPs. Thus, before forming a problem instance, each scenario is solved separately. Since our GA is used to solve these classical QAPs, results are not guaranteed to be optimal but still claimed to be close to optimal relying on the validation of the algorithm.

There are two types of problem instance settings. At the first setting, problem instances consist of scenarios having best objective values very far from one another. These are called problems with *dissimilar scenarios*. At the second setting, the situation is exactly the opposite and called problems with *similar scenarios*. Possessing problems with both dissimilar and similar scenarios enables us to observe how the performance of our formulations alters.

Six different problem sizes are selected for our experimentations, in parallel to the general problem sizes of problem instances available in QAPLIB. These are 12, 18, 20, 22, 25, and 30. Two problem instance groups with different problem sizes are obtained and the rest of the research is made using these problem instance groups, dissimilar scenarios and similar scenarios. Best objective values obtained from solving 36 classical QAPs are used in our problem instances are presented in Table 5.8 and 5.9. The range values reflects the types of the problem instances.

Table 5.8. Problem instances with dissimilar scenarios.

Problem Name	**Problem Size**	**High**	**Medium**	**Low**	**Range**
F3size12seed121_148	12	268184	185788	69746	**198438**
F3size18seed181_239	18	147438	112898	55854	**91584**
F3size20seed201_234	20	125382	61930	30608	**94774**
F3size22seed221_569	22	147382	68632	17460	**129922**
F3size25seed251_149	25	123666	77780	5978	**117688**
F3size30seed301_129	30	126910	78784	4654	**122256**

Table 5.9. Problem instances with similar scenarios.

Problem Name	**Problem Size**	**High**	**Medium**	**Low**	**Range**
F3size12seed121_259	12	159416	154222	152852	**6564**
F3size18seed181_578	18	122754	122516	117962	**4792**
F3size20seed201_169	20	80018	79838	76120	**3898**
F3size22seed221_124	22	108144	102408	91870	**16274**
F3size25seed251_567	25	66894	60888	57304	**9590**
F3size30seed301_358	30	61076	57926	54682	**6394**

These twelve problem instances are successfully solved by our GAs each based on different formulation created during this research to enhance different stochastic and robust performance measures.

5.3. Results and Comparison

Extensive numerical study enables us to achieve our purpose of comparing performances of different approaches in terms of robustness metrics and to gain important insights into ways of treating the uncertainty issue in facility layout problem.

It is important to recall that the classical QAPs corresponding to each scenario in our problem instances are solved beforehand since their optimal objective function

values are necessary for the formulations containing the regret criteria. The information generated during the GA search for the scenario solutions is exploited for initial population generation in the other GA for our models (i.e., the permutations of the scenario optimal are used as seed individual in the initial populations; the Pareto optimal solutions in the final population of scenario GAs are used in the initial population for the multi-objective cases.)

5.3.1. Results and Comparison of the First Four Models

At the initial stage of our analysis, results of the first four formulations are evaluated. All of these formulations have single objective functions and cover different robustness metrics. In the current and the following sections of this thesis, the names of the formulations used are "*expected cost*", "*minimax cost*", "*absolute minimax regret*" and "*relative minimax regret*".

The primary criterion in evaluation of the performances of different formulations is the objective values found in terms of all stochastic and robustness approaches. Each formulation is dedicated to a different measure and throughout their corresponding algorithms, the comparison of solutions are made based on their own measures. For instance, minimax cost algorithm efforts to attain individuals with least maximum cost values in their final populations.

Our analysis tries to capture whether each of these formulations are justifiable on their own right and whether they are distinct in behavior and leading us to different solutions where robustness is interpreted in a different way. In other words, we have targeted to find an answer to the question of how a solution proposed by one formulation behaves in other performance measures that are not originally addressed in the solution procedure of its formulation.

In order to construct our analysis, all 12 problems instance are solved with all of the formulations developed. To reduce the effect of randomness, solution procedures

are replicated 10 times and the best and average performances are evaluated.

The numerical results can be seen at the following tables. Each table is dedicated to a different performance measure. The comparison criterion is expected cost in Table 5.10, maximum cost in Table 5.11, maximum absolute regret in Table 5.12, and maximum relative regret in Table 5.13. The rows of the tables correspond to the results for each problem instance and the columns of the tables correspond to the results of each formulation. So in each column, the best solution's behavior is provided in terms of the criterion used in that table. Two columns are reserved for each formulation, one column for the best of ten replications and one for the average of ten replications. For instance, the expected cost values of the best solutions found by four formulations are provided in Table 5.10.

It is important to recall that only one of the formulations actually aims to performance measure used in that table. So, in Table 5.10, only the expected cost formulation's objective is to generate solutions with minimum expected cost values. Other formulations try to generate solutions based on their own performance measures and after their best solutions are obtained, expected cost values of those solutions are calculated and provided in Table 5.10.

In each row, best results among different formulations are bolded meaning that the corresponding formulation of the bolded cell outperforms the others in terms of the criterion of that table.

The problem instances are grouped in terms of whether they are composed of dissimilar or similar scenarios and then sorted based on their problem sizes. At the end of each problem instance group, one row is dedicated for the numbers displaying how many times a formulation provided the best results among all formulations to see whether there is any effect of a problem instance being composed of dissimilar or similar scenarios. High numbers mean the corresponding formulation performs usually better.

Table 5.10. Comparison of single objective formulations with Expected Cost criterion.

Criterion: *Expected Cost*	**Model**							
	Expected Cost		**Minimax Cost**		**Absolute Minimax Regret**		**Relative Minimax Regret**	
Problem Name	**Best**	**Average**	**Best**	**Average**	**Best**	**Average**	**Best**	**Average**
F3size12seed121_148	***192242.67***	***192242.67***	214684.00	214684.00	***192242.67***	***192242.67***	***192242.67***	***192242.67***
F3size18seed181_239	***137786.00***	***138010.27***	142847.33	145688.47	138684.00	140077.07	147300.67	147865.07
F3size20seed201_234	***97746.67***	***97746.67***	***97746.67***	***97746.67***	102822.00	105213.67	106224.67	113868.73
F3size22seed221_569	***103384.67***	***106241.27***	133949.33	116596.73	110193.33	112191.67	121852.67	125415.53
F3size25seed251_149	***82711.33***	***82711.33***	98428.00	98428.00	84736.00	85576.07	95505.33	101795.07
F3size30seed301_129	***91438.67***	***93573.07***	96776.67	101168.87	96490.67	97131.93	104400.67	108079.13
Variable Scenarios Total	***6***	***6***	***1***	***1***	***1***	***1***	***1***	***1***
F3size12seed121_259	***174822.67***	***174822.67***	***174822.67***	***174822.67***	***174822.67***	***174822.67***	***174822.67***	***174822.67***
F3size18seed181_578	***150128.00***	***151445.07***	151712.00	153858.87	151712.00	154242.00	154407.33	154553.73
F3size20seed201_169	***107368.67***	***108690.00***	110559.33	113700.60	110777.33	113026.47	108122.00	112842.67
F3size22seed221_124	***132754.00***	***136286.53***	138369.33	141189.73	135594.00	140885.40	135594.00	140862.00
F3size25seed251_567	***88706.67***	***91380.53***	92280.00	96262.67	91728.67	96442.20	90681.33	95262.47
F3size30seed301_358	***81842.00***	***84224.73***	83379.33	86171.47	82482.67	87638.67	83167.33	86564.27
Similar Scenarios Total	***6***	***6***	***1***	***1***	***1***	***1***	***1***	***1***
Total	***12***	***12***	***2***	***2***	***2***	***2***	***2***	***2***

Table 5.11. Comparison of single objective formulations with Maximum Cost criterion.

Criterion: *Maximum Cost*	**Model**							
	Expected Cost		**Minimax Cost**		**Absolute Minimax Regret**		**Relative Minimax Regret**	
Problem Name	**Best**	**Average**	**Best**	**Average**	**Best**	**Average**	**Best**	**Average**
F3size12seed121_148	289230	289230.00	***268184***	***268184.00***	289230	289230.00	289230	289230.00
F3size18seed181_239	183710	181322.60	***163372***	***164907.20***	179948	180645.20	205858	205898.00
F3size20seed201_234	***125382***	***125382.00***	***125382***	***125382.00***	155916	157100.60	178566	193891.80
F3size22seed221_569	157708	169824.60	***146488***	***147292.60***	180196	181807.20	234482	238856.60
F3size25seed251_149	145454	145454.00	***123994***	***123994.00***	139316	143200.60	176202	184860.40
F3size30seed301_129	165092	161657.00	***132664***	***136113.80***	153618	155973.80	190948	196670.00
Variable Scenarios Total	**1**	**1**	**6**	**6**	**0**	**0**	**0**	**0**
F3size12seed121_259	***180090***	***180090.00***	***180090***	***180090.00***	***180090***	***180090.00***	***180090***	***180090.00***
F3size18seed181_578	165644	160015.80	***154742***	***155946.00***	***154742***	157105.40	157246	157838.00
F3size20seed201_169	***110250***	122833.60	113212	***115484.60***	112414	116254.60	113266	116314.20
F3size22seed221_124	150250	149478.80	***139222***	***143363.60***	144484	148101.40	144484	148792.40
F3size25seed251_567	94890	104897.60	***94544***	***98358.60***	99268	102231.40	100620	105346.00
F3size30seed301_358	87122	94067.20	***85848***	***88308.40***	87448	91489.60	87928	92299.60
Similar Scenarios Total	**2**	**1**	**5**	**6**	**2**	**1**	**1**	**1**
Total	**3**	**2**	**11**	**12**	**2**	**1**	**1**	**1**

Table 5.12. Comparison of single objective formulations with Maximum Absolute Regret criterion.

Criterion: *Maximum Absolute Regret*	**Model**							
	Expected Cost		**Minimax Cost**		**Absolute Minimax Regret**		**Relative Minimax Regret**	
Problem Name	**Best**	**Average**	**Best**	**Average**	**Best**	**Average**	**Best**	**Average**
F3size12seed121_148	***23332***	***23332.00***	72552	72552.00	***23332***	***23332.00***	***23332***	***23332.00***
F3size18seed181_239	36564	38437.40	50474	54996.60	***35232***	***36773.00***	58420	58460.00
F3size20seed201_234	51816	51816.00	51816	51816.00	***31812***	***35018.60***	53184	68509.80
F3size22seed221_569	49108	39682.40	102596	72298.40	***33004***	***36302.60***	87100	91474.60
F3size25seed251_149	21788	21788.00	46214	46214.00	***16566***	***20058.80***	52536	61194.40
F3size30seed301_129	38182	35437.60	53880	56253.40	***26708***	***30433.20***	64038	69760.00
Variable Scenarios Total	**1**	**1**	**0**	**0**	**6**	**6**	**1**	**1**
F3size12seed121_259	***25868***	***25868.00***	***25868***	***25868.00***	***25868***	***25868.00***	***25868***	***25868.00***
F3size18seed181_578	43128	37680.40	***32226***	35229.80	***32226***	***34913.00***	34730	35371.60
F3size20seed201_169	34130	43219.00	33374	37175.60	***32576***	36545.00	33384	***36382.60***
F3size22seed221_124	47842	46989.80	47018	50282.00	***37398***	42473.60	***37398***	***42293.20***
F3size25seed251_567	***29672***	41159.80	33656	37902.40	32374	***36371.20***	33726	38452.00
F3size30seed301_358	29196	35121.80	27162	***30611.40***	***26372***	31254.00	26852	31223.60
Similar Scenarios Total	**2**	**1**	**2**	**2**	**5**	**3**	**2**	**3**
Total	**3**	**2**	**2**	**2**	**11**	**9**	**3**	**4**

Table 5.13. Comparison of single objective formulations with Maximum Relative Regret criterion.

Criterion: *Maximum Relative Regret*	**Model**							
	Expected Cost		**Minimax Cost**		**Absolute Minimax Regret**		**Relative Minimax Regret**	
Problem Name	**Best**	**Average**	**Best**	**Average**	**Best**	**Average**	**Best**	**Average**
F3size12seed121_148	***0.126***	***0.126***	0.685	0.685	***0.126***	***0.126***	***0.126***	***0.126***
F3size18seed181_239	0.655	0.633	0.842	0.955	0.575	0.615	***0.401***	***0.422***
F3size20seed201_234	0.837	0.837	0.837	0.837	1.039	1.047	***0.552***	***0.608***
F3size22seed221_569	2.813	2.141	5.876	4.141	1.792	2.026	***0.591***	***0.629***
F3size25seed251_149	3.165	3.165	6.967	6.967	2.771	3.118	***0.425***	***0.515***
F3size30seed301_129	3.300	2.826	5.510	6.209	5.580	4.989	***0.505***	***0.562***
Variable Scenarios Total	**1**	**1**	**0**	**0**	**1**	**1**	**6**	**6**
F3size12seed121_259	***0.168***	***0.168***	***0.168***	***0.168***	***0.168***	***0.168***	***0.168***	***0.168***
F3size18seed181_578	0.352	0.309	***0.263***	0.296	***0.263***	0.291	0.284	***0.290***
F3size20seed201_169	0.448	0.544	0.418	0.480	***0.416***	0.468	0.439	***0.460***
F3size22seed221_124	0.467	0.466	0.512	0.547	***0.365***	0.437	***0.365***	***0.420***
F3size25seed251_567	0.518	0.662	0.565	0.646	***0.495***	0.607	0.523	***0.583***
F3size30seed301_358	0.504	0.596	0.469	0.548	***0.437***	0.553	0.485	***0.525***
Similar Scenarios Total	**1**	**1**	**2**	**1**	**6**	**1**	**2**	**6**
Total	**2**	**2**	**2**	**1**	**7**	**2**	**8**	**12**

The results in the tables above reveal that each formulation leads up to solutions with distinct characteristics. Each formulation can only cover the measure that is dedicated and none of them can substitute the other. In order to obtain solutions with least expected cost values, the formulation that is dedicated to find expected cost optimum must be employed and the similar conclusion might be made for the rest of the formulations. Other formulations can obtain the best results in only a few problem instances and this is the case when problem sizes are small.

To conclude, we may say that no matter what type of problem instance group or problem size is selected these results are the same. All measures are meaningful but exclusive with one another. Decision maker needs to determine which robustness measure to optimize and then employ its corresponding algorithm.

The exception of these results exist for the absolute minimax regret and relative minimax regret formulations in the problem instances composing of similar scenarios. The reason is when the optimal objective function values of scenarios are close with each other, the constraints in the absolute minimax regret and relative minimax regret formulations where regret is defined become alike and their results become indistinct. So, in the cases where variability across the scenarios is not severe, two models can be used interchangeable.

Furthermore, the average time required in seconds for a single run of each algorithm is presented in Table 5.14 since time requirement may also be another comparison criterion. From the results, it can be claimed that there is no significant difference between execution times of algorithms.

Table 5.14. Average time spent (in seconds) during algorithms.

Problem Name	**Problem Size**	**Expected Cost**	**Minimax Cost**	**Absolute Minimax Regret**	**Relative Minimax Regret**
F3size12seed121_148	12	14,53	13,46	13,28	13,53
F3size12seed121_259	12	15,14	13,52	13,85	14,07
F3size18seed181_239	18	60,12	57,49	54,51	49,47
F3size18seed181_578	18	62,21	54,16	54,24	54,52
F3size20seed201_169	20	91,50	78,38	78,80	79,93
F3size20seed201_234	20	85,49	92,28	80,10	84,84
F3size22seed221_124	22	158,40	137,47	136,59	134,87
F3size22seed221_569	22	130,63	135,05	121,02	136,57
F3size25seed251_567	25	218,64	200,60	196,06	188,51
F3size25seed251_149	25	207,00	198,54	188,70	315,99
F3size30seed301_358	30	337,22	312,57	313,67	316,85
F3size30seed301_129	30	377,23	371,11	358,74	426,70

5.3.2. Results of p-Robustness Model

The p-robustness methodology aims to combine the advantages of both stochastic and robust optimization approaches by seeking the least-cost solution in the expected value that is p-robust. By determining an appropriate value for the p parameter, the one can achieve solutions performing well in expected cost and having maximum regret value less than this determined robustness parameter.

In order to determine the p values, first the behaviors of the solutions generated by expected cost and relative minimax regret models should be examined. The results for the former model might have the least expected cost values but higher maximum relative regret values. Similarly, the results for the latter model might have the least maximum relative regret values but higher expected cost values. If p is chosen to be

higher than the maximum relative regret values obtained from the expected cost model, the p-robustness constraint will be redundant as the solutions from the expected cost model already satisfies this constraint. Also, if p is chosen to be less than the maximum relative regret values obtained from the minimax relative regret model, there may be no feasible solutions available as the p-robustness constraint makes the problem infeasible.

Therefore, in our p-robustness analysis, there should be some significant amount of difference between the maximum relative regret results of two formulations and problems are solved with several levels for the p parameter. When we examine the results of the previous section, only three problem instances where problem size is large and problem instances are composed of dissimilar scenarios are suitable to make p-robustness analysis.

The results of the solutions obtained from p-robustness formulation are presented in Table 5.15, 5.16, and 5.17, each for one problem instance. Each row presents the results of the best solution obtained from ten replications where the first row is for expected cost model and the last row is for minimax relative regret model. Two related criteria which are expected cost criterion and maximum relative regret criterion are presented in two columns. It can be observed that as p value is scaled down in each row, solutions keep satisfying this maximum allowable regret value but their expected costs increase. If p takes smaller values the results obtained from minimax relative regret formulation, there may be no feasible p-robust solutions for the problem as it is observed in every problem instances and presented in tables.

Table 5.15. p-robustness Analysis on "*F3size22seed221_569*".

Model Name	**Expected Cost Criterion**	**Maximum Relative Regret**
Expected Cost	103384.67	2.813
p-robustness, p= 3	103384.67	2.813
p-robustness, p= 2	108593.67	1.876
p-robustness, p= 1	109382.00	0.907
p-robustness, p= 0.75	118343.67	0.720
p-robustness, p= 0.5	*no feasible solution*	
Relative Minimax Regret	121852.67	0.591

Table 5.16. p-robustness Analysis on "*F3size25seed251_149*".

Model Name	**Expected Cost Criterion**	**Maximum Relative Regret**
Expected Cost	82711.33	3.165
p-robustness, p= 5	82711.33	3.165
p-robustness, p= 3	87339.67	2.821
p-robustness, p= 1	90234.67	0.882
p-robustness, p= 0.5	95505.33	0.425
p-robustness, p= 0.4	*no feasible solution*	
Relative Minimax Regret	95505.33	0.425

Table 5.17. p-robustness Analysis on "*F3size30seed301_129*".

Model Name	**Expected Cost Criterion**	**Maximum Relative Regret**
Expected Cost	91438.67	3.300
p-robustness, p= 5	91438.67	3.300
p-robustness, p= 3	97922.67	2.890
p-robustness, p= 2	101160.33	1.504
p-robustness, p= 1	102398.67	0.967
p-robustness, p= 0.5	*no feasible solution*	
Relative Minimax Regret	104400.67	0.505

The behavior of solutions obtained from p-robustness formulation can also be observed in Figure 5.1, 5.2, and 5.3 where x-axis corresponds the expected cost values of solutions and y-axis corresponds maximum relative regret values. The results of p-robustness formulations with different p parameters are able to produce feasible solutions are plotted as well as the results of both expected cost and relative minimax regret formulations.

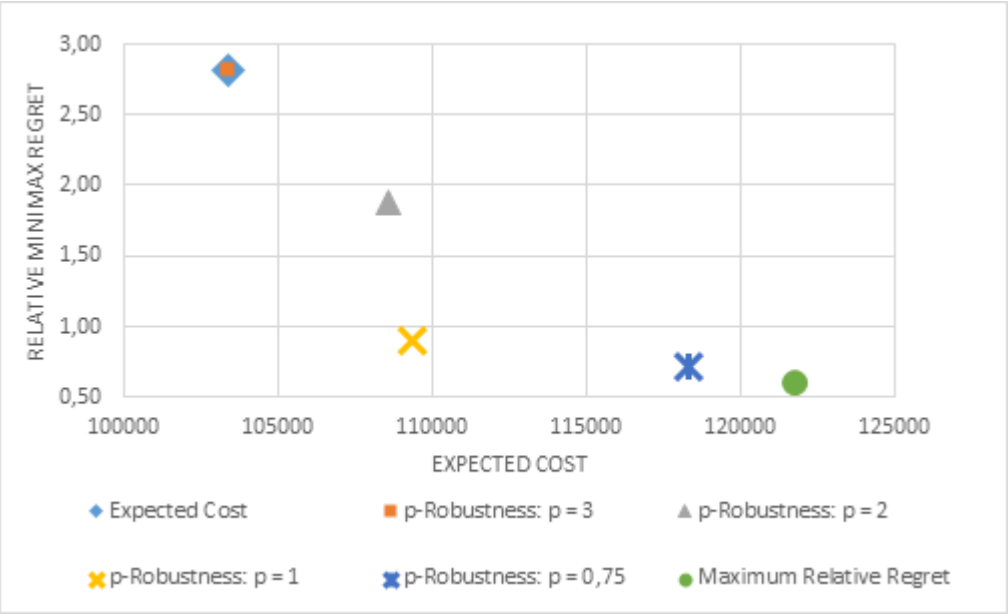

Figure 5.1. p-robustness results for problem instance "*F3size22seed221_569*".

Figure 5.2. p-robustness results for problem instance "*F3size25seed251_149*".

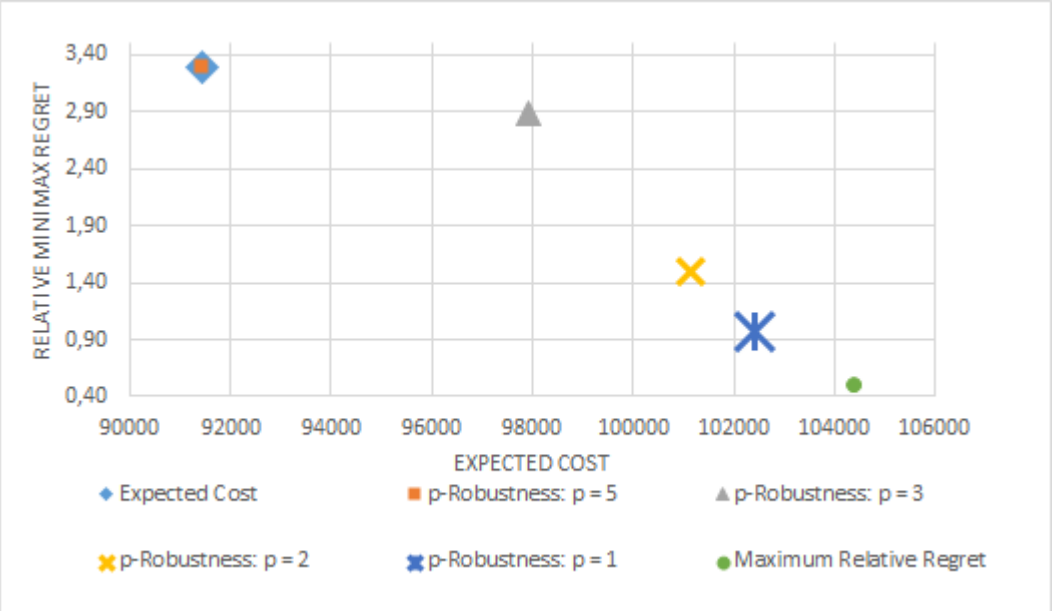

Figure 5.3. p-robustness results for problem instance "*F3size30seed301_129*".

The advantage of employing p-robustness formulation is that as a decision maker, we can select an appropriate value for p and obtain the least cost solution that satisfies the corresponding p-robustness constraint. This enables us to have control over the maximum regret value of the solutions thereby combine the benefits of the two criteria.

The drawback for p-robustness approach is that it is required to solve the same problem several times to obtain a set of p-robust solutions displaying the behavior with respect to p-robustness constraint. The time requirement of solving the problem for only a specified p value is almost equal is to solve the problem with the preceding formulations, so it is needed to wait for a much longer time to obtain valuable results from p-robustness approach.

5.3.3. Results of Expected Cost & Absolute Regret Model

Expected Cost & Absolute Regret formulation employs two objectives covered in the previous formulations in a collaborate manner and aims to obtain a set of feasible solutions that are satisfactory in terms of behaving well on average and being robust to all possible scenarios. With its multi-objective structure, we are able to produce the Pareto front in a single run and produce more information than the two separate runs corresponds to expected cost and absolute minimax regret formulations.

In our analysis, the final populations of 10 replications of the multi-objective GA are combined and then sorted based on the non-dominated sorting criteria to produce a single Pareto front. The solutions in the Pareto front obtained for one problem instance are represented by 2-dimensional vectors composed of expected cost and maximum absolute regret values in Figure 5.6 where x-axis corresponds the expected cost values of solutions and y-axis corresponds maximum absolute regret values. The solutions found by the related single objective formulations are also added to this plot. It is important to recall that the front obtained from our multi-objective genetic algorithm is an approximate Pareto front and based on the performance of our algorithm, there may be other non-dominated solutions not presented in this plot.

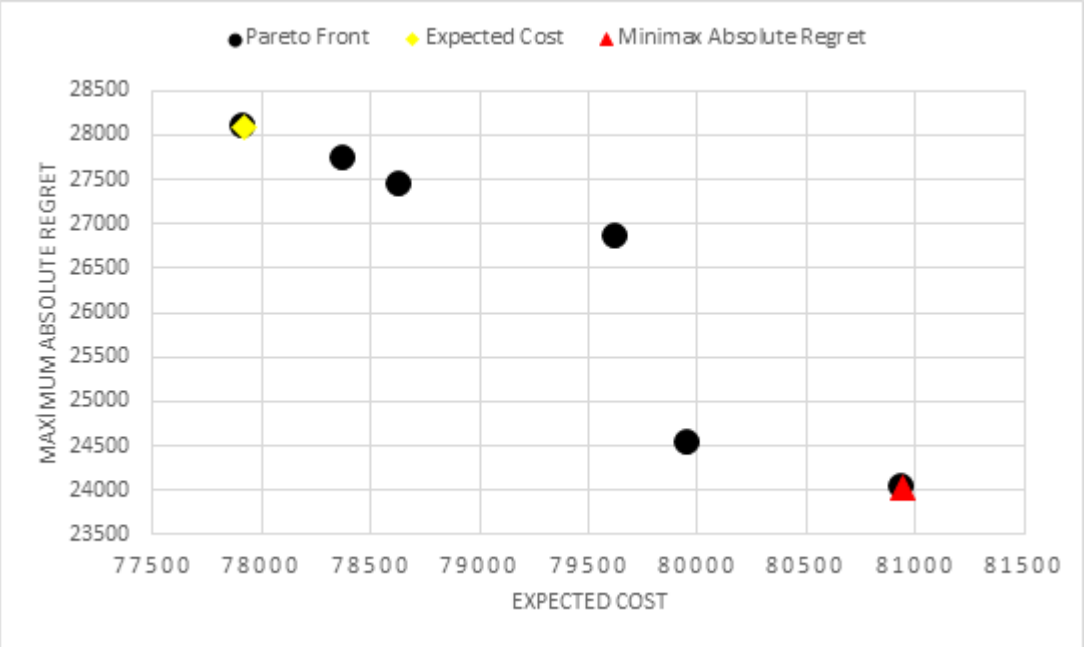

Figure 5.4. Expected Cost & Absolute Regret results for problem instance "*F3size30seed301_358*".

The detailed numerical results with the expected cost and maximum absolute regret comparison criteria are presented in Table 5.18 as this multi-objective formulation aims to find good solutions based on these two measures. Similar with the analysis made on single objective formulations, the rows of the tables correspond to the results in each problem instance and the columns of the tables correspond to the results of each formulation.

Best solutions found in Pareto front are compared with solutions proposed by single objective formulations. For instance, the solution proposed by expected cost formulation is compared with the results of the member of the Pareto front having the least expected cost value in Table 5.18. In this way, we can evaluate the performance of our multi-objective algorithm in terms of finding as good solutions as the single objective formulations.

Additionally, Tables 5.19 and 5.20 also present the comparison of the members of the Pareto front and the best solutions proposed by single objective formulations

Table 5.18. Expected Cost & Absolute Regret Analysis.

Criterion : ***Expected Cost***	**Model**		**Criterion :** ***Maximum Absolute Regret***	**Model**	
	Expected Cost	**EXP & ABS**		**Absolute Minimax Regret**	**EXP & ABS**
Problem Name	***Best***	***Best***	***Problem Name***	***Best***	***Best***
F3size12seed121_148	***192242.67***	***192242.67***	F3size12seed121_148	***23332***	***23332***
F3size18seed181_239	***137786.00***	***137786.00***	F3size18seed181_239	***35232***	***35232***
F3size20seed201_234	***97746.67***	***97746.67***	F3size20seed201_234	31812	***27898***
F3size22seed221_569	***103384.67***	***103384.67***	F3size22seed221_569	33004	***31744***
F3size25seed251_149	***82711.33***	***82711.33***	F3size25seed251_149	***16566***	16780
F3size30seed301_129	91438.67	***88492.67***	F3size30seed301_129	26708	***19864***
Dissimilar Scenarios Total	***5***	***6***	***Dissimilar Scenarios Total***	**3**	***5***
F3size12seed121_259	***174822.67***	***174822.67***	F3size12seed121_259	***25868***	***25868***
F3size18seed181_578	***150128.00***	***150128.00***	F3size18seed181_578	***32226***	***32226***
F3size20seed201_169	107368.67	***105792.67***	F3size20seed201_169	32576	***31636***
F3size22seed221_124	***132754.00***	***132754.00***	F3size22seed221_124	37398	***36630***
F3size25seed251_567	***88706.67***	91282.67	F3size25seed251_567	***32374***	32652
F3size30seed301_358	***77920.00***	***77920.00***	F3size30seed301_358	***24032***	***24032***
Similar Scenarios Total	***5***	***5***	***Similar Scenarios Total***	**4**	***5***
Total	***10***	***11***	***Total***	**7**	***10***

each dedicated to a different performance measure. Each column in Table 5.19 presents the results of a problem instance composed of dissimilar scenarios and each column in Table 5.20 presents the results of a problem instance composed of similar scenarios.

Table 5.19. Expected Cost & Absolute Regret Analysis: *dissimilar scenarios*.

Comparison Criteria	**Dissimilar Scenario Sizes**					
	12	**18**	**20**	**22**	**25**	**30**
Expected Cost	1.00	1.00	1.00	1.00	1.00	0.97
Maximum Cost	1.08	1.10	1.00	1.08	1.13	1.11
Maximum Absolute Regret	1.00	1.00	0.88	0.96	1.01	0.74
Maximum Relative Regret	1.00	1.44	1.32	2.68	4.96	6.82

The values are the ratios of the performances of the multi-objective algorithm over the single objective ones. A value less than one indicates that the multi-objective algorithm is capable of finding a member in its Pareto front such that the member performs better than the solutions proposed by the single objective formulations. The ratio equal to one means both formulations lead up with the same solutions and the ratio greater than one means there is no such solution in the Pareto front that performs better than the solutions proposed by the single objective formulations.

Table 5.20. Expected Cost & Absolute Regret Analysis: *similar scenarios*.

Comparison Criteria	**Similar Scenario Sizes**					
	12	**18**	**20**	**22**	**25**	**30**
Expected Cost	1.00	1.00	0.99	1.00	1.03	0.95
Maximum Cost	1.00	1.00	0.97	1.00	1.04	0.96
Maximum Absolute Regret	1.00	1.00	0.97	0.98	1.01	0.91
Maximum Relative Regret	1.00	0.93	0.90	1.08	0.99	0.87

The ratios in the first and the third rows in Table 5.19 and 5.20 are usually less than or equal to one, or close to one at least. This means, the multi-objective algorithm

is capable of finding as good solutions as the single objective ones. The ratios in the second and the fourth rows might take slightly large values indicating the Pareto front found by our multi-objective algorithm does not include solutions performing well under those measure. This fact also supports the results found in Section 5.3.1. As it has been mentioned, the performance measures cannot substitute each other and a formulation whose objective does not include minimax cost and minimax relative regret objectives to perform well in the corresponding measures.

An observation regarding the expected cost and minimax absolute regret formulation is that its front includes only a few number of solutions. As in Figure 5.6, there are only six members including two solution can also be found by single objective formulations and this case is more apparent when the problem size is smaller. For instance, when the optimum solution for expected cost and maximum absolute regret measures are the same solution, the corresponding Pareto front is composed of only one solution.

The reason of lack of members is that the formulation has two objective hence the front is 2-dimensional. If there were more than two objective, it would be harder for a solution to be dominated by another solution since dominating requires an improvement in all objectives. The other reason is the choice of the objectives. Both objectives are composed of minimizing a performance measure and even they do not serve the same measure they might be highly correlated. For instance, a solution which has a low expected cost value is likely to have a low maximum absolute regret value too. This correlation may be the primary reason that lacks the number of non-dominated solutions.

5.3.4. Results of mQAP Model

The case is quite different when we want to investigate the results of the multi-objective QAP formulation that include three objective functions as the number of scenarios in our problem instances are equal to three. We want to evaluate the per-

formance of multiple solutions located at the same front. The results are shown in MATLAB with version R2012a.

In Figure 5.5 and 5.6, the Pareto fronts obtained from our multi-objective genetic algorithm in two problem instances are presented as well as the solutions proposed by single objective formulations in order to make a visual comparison. The bowl shapes of the fronts can be detected.

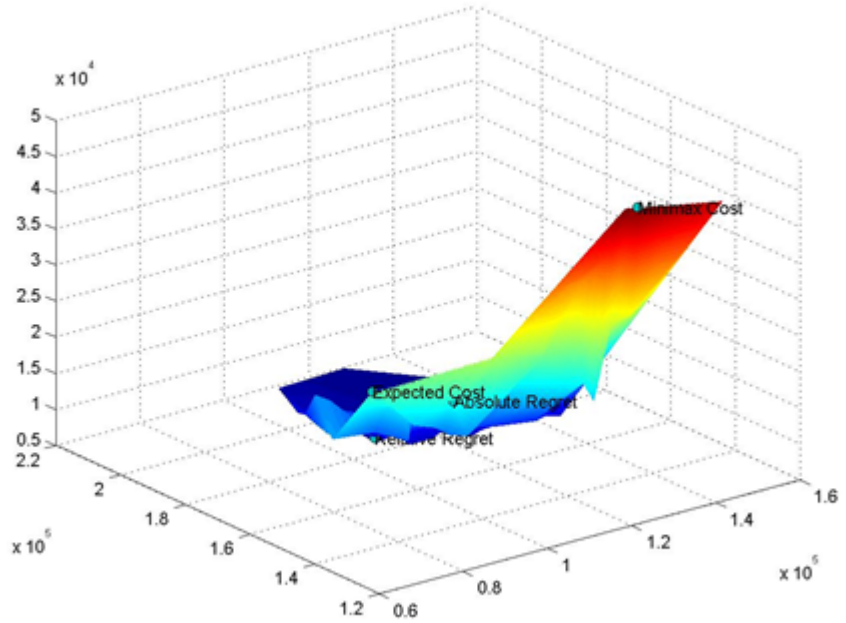

Figure 5.5. mQAP front for problem instance "*F3size25seed251_149*".

If solutions generated by single objective formulations reside on the front, it means that our multi-objective algorithm is capable of finding as good solutions as the single objective formulations have found for the corresponding performance measure. If solutions generated by single objective formulations are above the front, it means that our multi-objective algorithm finds better solutions than the single objective formulations. Finally, if solutions generated by single objective formulations are below the front, it means that our multi-objective algorithm missed some members of the true Pareto front.

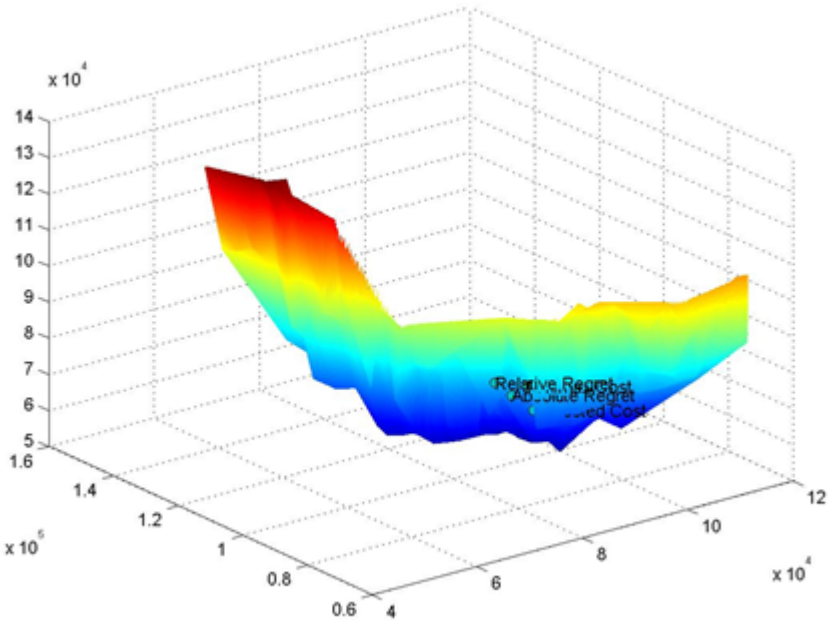

Figure 5.6. mQAP front for problem instance "*F3size30seed301_358*".

The comparison can also be made through the detailed numerical results that are presented in Table 5.21, and 5.22. Similar with the analysis made in Section 5.3.1, the columns of the tables represents the results of each formulation and the rows of the tables displays the results in each problem instance.

In each comparison, the solutions proposed by single objective formulation are compared with a distinct member of Pareto front proposed by mQAP formulation. The aim here is to show to that our multi-objective genetic algorithm is capable of finding the members of true Pareto front and while doing so, it is capable of finding as many members as possible in order not to miss any member which may represent the optimal solution for a performance measure we work.

The comparison results indicates that our mQAP formulations is adequate to find the optimal solutions. Tables 5.23 and 5.24 also present the comparison of the members of the Pareto front and the best solutions proposed by single objective formulations. The values are again the ratios of the performances of the multi-objective algorithm over the single objective ones.

Table 5.21. mQAP Analysis for Expected and Maximum Cost.

Criterion : ***Expected Cost***	**Model**		**Criterion :** ***Maximum Cost***	**Model**	
	Expected Cost	**mQAP**		**Minimax Cost**	**mQAP**
Problem Name	***Best***	***Best***	***Problem Name***	***Best***	***Best***
F3size12seed121_148	***192242.67***	***192242.67***	F3size12seed121_148	***268184***	***268184***
F3size18seed181_239	***137786.00***	140092.00	F3size18seed181_239	***163372***	165696
F3size20seed201_234	***97746.67***	***97746.67***	F3size20seed201_234	***125382***	***125382***
F3size22seed221_569	***103384.67***	***103384.67***	F3size22seed221_569	***146488***	147382
F3size25seed251_149	82711.33	***81197.33***	F3size25seed251_149	***123994***	***123994***
F3size30seed301_129	91438.67	***86802.00***	F3size30seed301_129	***132664***	***132664***
Dissimilar Scenarios Total	***4***	***5***	***Dissimilar Scenarios Total***	***6***	***4***
F3size12seed121_259	***174822.67***	***174822.67***	F3size12seed121_259	***180090***	***180090***
F3size18seed181_578	***150128.00***	151712.00	F3size18seed181_578	***154742***	***154742***
F3size20seed201_169	107368.67	***105792.67***	F3size20seed201_169	113212	***111474***
F3size22seed221_124	***132754.00***	136452.67	F3size22seed221_124	***139222***	143796
F3size25seed251_567	***88706.67***	91219.33	F3size25seed251_567	***94544***	98246
F3size30seed301_358	81842.00	***79364.00***	F3size30seed301_358	***85848***	86136
Similar Scenarios Total	***4***	***3***	***Similar Scenarios Total***	**5**	**3**
Total	***8***	***8***	***Total***	**11**	**7**

Table 5.22. mQAP Analysis for Maximum Absolute and Relative Regret.

Criterion : *Maximum Absolute Regret*	**Model**		**Criterion : *Maximum Relative Regret***	**Model**	
	Absolute Minimax Regret	**mQAP**		**Relative Minimax Regret**	**mQAP**
Problem Name	***Best***	***Best***	***Problem Name***	***Best***	***Best***
F3size12seed121_148	***23332***	***23332***	F3size12seed121_148	***0.126***	***0.126***
F3size18seed181_239	***35232***	38676	F3size18seed181_239	***0.401***	0.432
F3size20seed201_234	***31812***	31942	F3size20seed201_234	0.552	***0.536***
F3size22seed221_569	***33004***	33052	F3size22seed221_569	***0.591***	0.636
F3size25seed251_149	***16566***	***16566***	F3size25seed251_149	0.425	***0.403***
F3size30seed301_129	26708	***17768***	F3size30seed301_129	0.505	***0.451***
Dissimilar Scenarios Total	**5**	***3***	***Dissimilar Scenarios Total***	**3**	***4***
F3size12seed121_259	***25868***	***25868***	F3size12seed121_259	***0.168***	***0.168***
F3size18seed181_578	***32226***	***32226***	F3size18seed181_578	0.284	***0.263***
F3size20seed201_169	32576	***31636***	F3size20seed201_169	0.439	***0.396***
F3size22seed221_124	***37398***	39490	F3size22seed221_124	***0.365***	0.382
F3size25seed251_567	***32374***	34046	F3size25seed251_567	***0.523***	0.568
F3size30seed301_358	***26372***	28210	F3size30seed301_358	***0.485***	0.487
Similar Scenarios Total	**5**	***3***	***Similar Scenarios Total***	**4**	***3***
Total	**10**	***6***	***Total***	**7**	***7***

Unlike the analysis in the previous section, for evaluating the quality of our multi-objective genetic algorithm, it is required to examine the ratios in all rows. The ratios are usually less than or equal to one or at least close to one. This indicates that our algorithm performs as well as the single objective algorithms. The one can construct the Pareto front and then by evaluating only the members of this front, he can find a fulfilling solution based on his requirements.

Table 5.23. mQAP Analysis: *dissimilar scenarios.*

Comparison Criteria	**Dissimilar Scenario Sizes**					
	12	**18**	**20**	**22**	**25**	**30**
Expected Cost	1.00	1.02	1.00	1.00	0.98	0.95
Maximum Cost	1.00	1.01	1.00	1.01	1.00	1.00
Maximum Absolute Regret	1.00	1.10	1.004	1.001	1.00	0.67
Maximum Relative Regret	1.00	1.08	0.97	1.08	0.95	0.89

Table 5.24. mQAP Analysis: *similar scenarios.*

Comparison Criteria	**Similar Scenario Sizes**					
	12	**18**	**20**	**22**	**25**	**30**
Expected Cost	1.00	1.01	0.99	1.03	1.03	0.97
Maximum Cost	1.00	1.00	0.98	1.03	1.04	1.003
Maximum Absolute Regret	1.00	1.00	0.97	1.06	1.05	1.07
Maximum Relative Regret	1.00	0.93	0.90	1.04	1.09	1.004

What's more, the time requirement for our multi-objective algorithm is the same as the time requirements of the single objective algorithms. Instead of solving the problems several times each for a different performance measure, we can solve the multi-objective algorithm and obtain the Pareto front which is composed of good compromises between the scenario objectives and include the optimal solutions for every performance measure.

On the other hand, the ratios that are greater than one indicates that we did not manage to find all members of the true Pareto front or the front we obtained includes dominated solutions. This is the case especially for the problems composing of similar scenarios and with large sizes. To investigate the reasons behind, some further analysis is required. As the Pareto fronts are formed by combining the fronts of ten replications, the number of solutions in the combined Pareto front might be significantly larger than the population size parameter applied in our multi-objective genetic algorithm. this indicates that, a single run of the algorithm is not capable of forming the entire Pareto front due to its population size parameter. To observe the effect of the population size parameter, experimentations with population size 200 are held and the sizes of the resulting Pareto front are summarized in Table 5.25.

Table 5.25. Front sizes for different population size levels.

Problem Name	**Population Size**	
	100	**200**
F3size12seed121_148	57	57
F3size12seed121_259	80	81
F3size18seed181_239	182	243
F3size18seed181_578	162	196
F3size20seed201_234	162	200
F3size20seed201_169	147	154
F3size22seed221_569	208	262
F3size22seed221_124	144	178
F3size25seed251_149	134	164
F3size25seed251_567	170	215
F3size30seed301_129	152	216
F3size30seed301_358	222	319

Tables 5.26 and 5.27 present the updates ratios for the comparison of the members of the Pareto front and the best solutions proposed by single objective formulations. The values are again the ratios of the performances of the multi-objective algorithm over

the single objective ones and the results indicate that there is a significant improvement by the multi-objective algorithm as the ratios are usually less than or equal to one. On the other hand, increase in the population size of our multi-objective algorithm increases the time requirement.

Table 5.26. mQAP Analysis: *dissimilar scenarios* with population size: 200.

Comparison Criteria	**Dissimilar Scenario Sizes**					
	12	**18**	**20**	**22**	**25**	**30**
Expected Cost	1.00	1.00	1.00	1.00	0.98	0.94
Maximum Cost	1.00	1.01	1.00	1.01	1.00	1.00
Maximum Absolute Regret	1.00	1.00	0.88	0.93	1.07	0.99
Maximum Relative Regret	1.00	1.00	0.97	1.05	0.95	0.81

Table 5.27. mQAP Analysis: *similar scenarios* with population size: 200.

Comparison Criteria	**Similar Scenario Sizes**					
	12	**18**	**20**	**22**	**25**	**30**
Expected Cost	1.00	1.01	0.99	1.03	1.03	0.95
Maximum Cost	1.00	1.00	0.98	1.00	0.99	0.98
Maximum Absolute Regret	1.00	1.00	0.97	0.98	0.99	0.99
Maximum Relative Regret	1.00	0.93	0.90	1.04	1.03	0.89

It can be concluded that to obtain satisfactory results from the multi-objective algorithm, population size parameter should be determined large enough to approximate the true Pareto front successfully. As the problem size increases, the requirement for the population size parameter may be large.

Another aspect that affects the performance of our multi-objective genetic algorithm is number of scenarios in the problem instances. It is clear that as the number of scenarios increases the count of the solutions in the true Pareto front increases and the requirement for the population size will also increase. to observe this effect, we

have made experimentations by combining scenarios to build new problem instances composed of 6 scenarios. each problem instance is solved successfully by our single objective algorithms and the then solved by mQAP formulation with population size equal to 1000. The ratios of the performances are presented in Table 5.28.

Table 5.28. mQAP Analysis: problem instances with 6 scenarios.

Comparison Criteria	**Problem Sizes**					
	12	**18**	**20**	**22**	**25**	**30**
Expected Cost	1.00	1.00	1.01	1.01	1.01	1.02
Maximum Cost	1.00	1.003	1.03	1.00	1.02	1.04
Maximum Absolute Regret	1.13	1.05	1.00	1.22	1.27	1.32
Maximum Relative Regret	1.00	1.01	1.03	1.06	1.05	1.08

Although the time requirement becomes very large compared to the ones with single objectives, the results indicate there is still need for increasing the population size parameter to obtain a good representative set of the true Pareto front. It is clear that additional work is required for the proposed multi-objective genetic algorithm to solve problem instances with large number of scenarios effectively and in reasonable times.

6. CONCLUSION

In this thesis, the aim is to investigate different ways of dealing with uncertainty to design a facility layout which attains robust and efficient performance under all possible scenarios. The QAP is selected as our benchmark problem and we formulate seven QAP-based formulations each covers different stochastic and robustness performance measures. Proposed formulations are solved using a GA with operators and local improvement schemes specially selected and adapted from literature. To the best of our knowledge, QAP has never been investigated in this context previously.

We also propose a multi-objective genetic algorithm which is similar to NSGA-II introduced by Deb *et al.* (2002) to approximate the Pareto front in our multi-objective formulations. The motivation behind developing multi-objective formulations comes from the strong relationship between robustness measures and the multi-objective formulations as pointed out by Aissi *et al.* (2009). Instead of evaluating each performance measure separately we can approximate the Pareto front in a single algorithm dedicated to mQAP formulation and select the appropriate member based on our chosen performance measure.

After the experimental setting, fine tuning and the validation for our proposed GA, problem instances of different sizes and characteristic are generated in order to evaluate the performances of the formulations developed. Extensive numerical analysis enables us to compare the performance of these approaches in terms of robustness metrics and to gain important insights into ways of treating the uncertainty issue in facility layout problem. We find that each robustness measure we discussed are distinct in behavior and cannot substitute one another leading us to different solutions where robustness is interpreted in a different way. Furthermore, we observe that the proposed GA procedure is capable of finding very good solutions to these problem.

The approximated Pareto front generated by the mQAP formulation offers the

decision maker a plenty of good quality solutions that include the optimal solutions for all robustness measures covered in this thesis. In fact, in the three scenarios case, the MOEA adapted from NSGA-II outperforms the single objective robust optimization GAs on most of the instances. This is a very promising result which was not previously investigated in the literature. Furthermore, it is also useful in many real life cases, since for many decision makers it might be natural and convenient to generate only three scenarios to represent the uncertainty in demand. However, as the problem gets larger in terms of its size and the number of scenarios, it becomes difficult to obtain close approximation of the Pareto front and represents the whole range of it.

As a future research topic, it is clear that additional work is required to the proposed multi-objective genetic algorithm to solve problem in instances with large number of scenarios efficiently and in reasonable times. Considering the results found in this thesis, it will be a noteworthy attempt to draw out the properties of the extended problem instances in depth, such as discovering the size and landscape of the Pareto front and developing appropriate search strategies.

Finally, the observation obtained about the potential of the multi-objective version for the QAP may be generalized to combinatorial optimization problems. It is hoped that this study will stimulate further investigation in this topic.

REFERENCES

Ahuja, R. K., J. B. Orlin, and A. Tiwari, 2000, "A Greedy Genetic Algorithm for the Quadratic Assignment Problem", *Computers & Operations Research*, Vol. 27, No. 10, pp. 917-934.

Aissi, H., C. Bazgan, and D. Vanderpooten, 2009, "Min–max and Min–max Regret Versions of Combinatorial Optimization Problems: A survey", *European Journal of Operational Research*, Vol. 197, No. 2, pp. 427-438.

Benjaafar, S., and M. Sheikhzadeh, 2000, "Design of Flexible Plant Layouts", *IIE Transactions*, Vol. 32, No. 4, pp. 309-322.

Ben-Tal, A., L. El Ghaoui, and A. Nemirovski, 2009, "Robust Optimization", Princeton University Press, New Jersey.

Beyer, H., and B. Sendhoff, 2007, "Robust Optimization–A Comprehensive Survey", *Computer Methods in Applied Mechanics and Engineering*, Vol. 196, No. 33, pp. 3190-3218.

Burkard, R. E., S. E. Karisch, and F. Rendl, 1997, "QAPLIB–A Quadratic Assignment Problem Library", *Journal of Global Optimization*, Vol. 10, No. 4, pp. 391-403.

Chen, G., M. S. Daskin, Z. M. Shen, and S. Uryasev, 2006, "The α-reliable Mean Excess Regret Model for Stochastic Facility Location Modeling", *Naval Research Logistics (NRL)*, Vol. 53, No. 7, pp. 617-626.

Coello, C. A., 2000, "An Updated Survey of GA-based Multi-objective Optimization Techniques", *ACM Computing Surveys (CSUR)*, Vol. 32, No. 2, pp. 109-143.

Coello, C. A. C., and A. D. Christiansen, 1998, "Two New GA-based Methods for Multi-objective Optimization", *Civil Engineering Systems*, Vol. 15, No. 3, pp. 207-243.

Coello, C. A. C., and G. B. Lamont, 2004, "Applications of Multi-objective Evolutionary Algorithms", World Scientific, New Jersey.

Coello, C. C., G. B. Lamont, and D. A. Van Veldhuizen, 2007, "Evolutionary Algorithms for Solving Multi-objective Problems", Springer, New York.

Daskin, M. S., S. M. Hesse, and C. S. Revelle, 1997, "α-reliable p-minimax Regret: A New Model for Strategic Facility Location Modeling", *Location Science*, Vol. 5, No. 4, pp. 227-246.

Davis, L., 1985, "Applying Adaptive Algorithms to Epistatic Domains", *IJCAI*, Vol. 85, pp. 162-164.

Day, R. O., and G. B. Lamont, 2005, "Multi-objective Quadratic Assignment Problem Solved by an Explicit Building Block Search Algorithm–MOMGA-IIa", In: *Evolutionary Computation in Combinatorial Optimization*, Springer.

Deb, K., 2001, "Multi-objective Optimization Using Evolutionary Algorithms", Vol. 16, John Wiley & Sons, New York.

Deb, K., and H. Gupta, 2005, "Searching for Robust Pareto-optimal Solutions in Multi-objective Optimization", *Lecture Notes in Computer Science*, Vol. 34, No. 10, pp. 150-164.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan, 2002, "A Fast and Elitist Multi-objective Genetic Algorithm: NSGA-II", *Evolutionary Computation, IEEE Transactions On*, Vol. 6, No. 2, pp. 182-197.

Drezner, Z., 2005, "Compounded Genetic Algorithms for the Quadratic Assignment Problem", *Operations Research Letters*, Vol. 33, No. 5, pp. 475-480.

Drezner, Z., 2008, "Extensive Experiments with Hybrid Genetic Algorithms for the Solution of the Quadratic Assignment Problem", *Computers & Operations Research*, Vol. 35, No. 3, pp. 717-736.

Gen, M., and R. Cheng, 1996, "A Survey of Penalty Techniques in Genetic Algorithms", In: *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*, IEEE.

Glover, F., and M. Laguna, 1999, "Tabu Search", Springer, New York.

Holland, J. H., 1975, "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.", U Michigan Press, Michigan.

Iancu, D. A., and N. Trichakis, 2013, "Pareto Efficiency in Robust Optimization", *Management Science*, Vol. 60, No. 1, pp. 130-147.

Jones, D. F., S. K. Mirrazavi, and M. Tamiz, 2002, "Multi-objective Meta-heuristics: An Overview of the Current State-of-the-art", *European Journal of Operational Research*, Vol. 137, No. 1, pp. 1-9.

Józefczyk, J., and M. Siepak, 2013, "Worst-case Regret Algorithms for Selected Optimization Problems with Interval Uncertainty", *Kybernetes*, Vol. 42, No. 3, pp. 371-382.

Kleeman, M. P., R. O. Day, and G. B. Lamont, 2004, "Analysis of a Parallel MOEA Solving the Multi-objective Quadratic Assignment Problem", In: *Genetic and Evolutionary Computation–GECCO 2004*, Springer.

Knowles, J. D., and D. Corne, 2002, "Towards Landscape Analyses to Inform the Design of Hybrid Local Search for the Multi-objective Quadratic Assignment Problem.", In: *HIS*.

Knowles, J., and D. Corne, 2003, "Instance Generators and Test Suites for the Multi-objective Quadratic Assignment Problem", In: *Evolutionary Multi-criterion Optimization*, Springer.

Konak, A., D. W. Coit, and A. E. Smith, 2006, "Multi-objective Optimization Using Genetic Algorithms: A Tutorial", *Reliability Engineering & System Safety*, Vol. 91, No. 9, pp. 992-1007.

Kouvelis, P., and G. Yu, 1997, "Robust Discrete Optimization and Its Applications", Springer, New York.

Kouvelis, P., A. A. Kurawarwala, and G. J. Gutierrez, 1992, "Algorithms for Robust Single and Multiple Period Layout Planning for Manufacturing Systems", *European Journal of Operational Research*, Vol. 63, No. 2, pp. 287-303.

Li, Y., P. M. Pardalos, and M. G. Resende, 1994, "A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem", *Quadratic Assignment and Related Problems*, Vol. 16, pp. 237-261.

Li, M., S. Azarm, and V. Aute, 2005, "A Multi-objective Genetic Algorithm for Robust Design Optimization", In: *Proceedings of the 2005 conference on Genetic and evolutionary computation*, ACM.

Lim, G. J., and A. D. Sonmez, 2013, "γ-Robust Facility Relocation Problem", *European Journal of Operational Research*, Vol. 229, No. 1, pp. 67-74.

Liu, Z., S. Guo, L. V. Snyder, A. Lim, and P. Peng, 2010, "A p-robust Capacitated Network Design Model with Facility Disruptions", In: *Advanced Manufacturing and Sustainable Logistics*, Springer.

Loiola, E. M., N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, 2007, "A Survey for the Quadratic Assignment Problem", *European Journal of Operational Research*, Vol. 176, No. 2, pp. 657-690.

López-Ibánez, M., L. Paquete, and T. Stützle, 2004, "On the Design of ACO for the Biobjective Quadratic Assignment Problem", In: *Ant Colony Optimization and Swarm Intelligence*, Springer.

Marler, R. T., and J. S. Arora, 2004, "Survey of Multi-objective Optimization Methods for Engineering", *Structural and Multidisciplinary Optimization*, Vol. 26, No. 6, pp. 369-395.

Mausser, H. E., and M. Laguna, 1998, "A New Mixed Integer Formulation for the Maximum Regret Problem", *International Transactions in Operational Research*, Vol. 5, No. 5, pp. 389-403.

Mausser, H. E., and M. Laguna, 1999, "Minimising the Maximum Relative Regret for Linear Programmes with Interval Objective Function Coefficients", *Journal of The Operational Research Society*, pp. 1063-1070.

Mausser, H. E., and M. Laguna, 1999, "A Heuristic to Minimax Absolute Regret for Linear Programs with Interval Objective Function Coefficients", *European Journal of Operational Research*, Vol. 117, No. 1, pp. 157-174.

Misevicius, A., 2003, "Genetic Algorithm Hybridized with Ruin and Recreate Procedure: Application to the Quadratic Assignment Problem", *Knowledge-Based Systems*, Vol. 16, No. 5, pp. 261-268.

Misevicius, A., 2004, "An Improved Hybrid Genetic Algorithm: New Results for the Quadratic Assignment Problem", *Knowledge-Based Systems*, Vol. 17, No. 2, pp. 65-73.

Norman, B. A., and A. E. Smith, 2006, "A Continuous Approach to Considering Uncertainty in Facility Design", *Computers & Operations Research*, Vol. 33, No. 6, pp. 1760-1775.

Owen, S. H., and M. S. Daskin, 1998, "Strategic Facility Location: A Review", *European Journal of Operational Research*, Vol. 111, No. 3, pp. 423-447.

Paquete, L., and T. Stützle, 2006, "A Study of Stochastic Local Search Algorithms for the Bi-objective QAP with Correlated Flow Matrices", *European Journal of Operational Research*, Vol. 169, No. 3, pp. 943-959.

Pardalos, L., and M. Resende, 1994, "A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem", *Discrete Mathematics and Theoretical Computer Science*, Vol. 16, pp. 237-261.

Peng, P., L. V. Snyder, A. Lim, and Z. Liu, 2011, "Reliable Logistics Networks Design with Facility Disruptions", *Transportation Research Part B: Methodological*, Vol. 45, No. 8, pp. 1190-1211.

Pierreval, H., and M. Plaquin, 1998, "An Evolutionary Approach of Multicriteria Manufacturing Cell Formation", *International Transactions in Operational Research*, Vol. 5, No. 1, pp. 13-25.

Quaranta, A. G., and A. Zaffaroni, 2008, "Robust Optimization of Conditional Value at Risk and Portfolio Selection", *Journal of Banking & Finance*, Vol. 32, No. 10, pp. 2046-2056.

Resende, M. G., K. Ramakrishnan, and Z. Drezner, 1995, "Computing Lower Bounds for the Quadratic Assignment Problem with an Interior Point Algorithm for Linear Programming", *Operations Research*, Vol. 43, No. 5, pp. 781-791.

Rosenblatt, M. J., and H. L. Lee, 1987, "A Robustness Approach to Facilities Design", *International Journal of Production Research*, Vol. 25, No. 4, pp. 479-486.

Sahni, S., and T. Gonzalez, 1976, "P-complete Approximation Problems", *Journal of The ACM (JACM)*, Vol. 23, No. 3, pp. 555-565.

Serra, D., and V. Marianov, 1998, "The p-median Problem in a Changing Network: The Case of Barcelona", *Location Science*, Vol. 6, No. 1, pp. 383-394.

Shapiro, A., and D. Dentcheva, 2009, "Lectures on Stochastic Programming: Modeling and Theory", Vol. 9, SIAM.

Snyder, L. V., 2006, "Facility Location Under Uncertainty: A Review", *IIE Transactions*, Vol. 38, No. 7, pp. 547-564.

Snyder, L. V., and M. S. Daskin, 2006, "Stochastic p-robust Location Problems", *IIE Transactions*, Vol. 38, No. 11, pp. 971-985.

Srinivas, N., and K. Deb, 1994, "Muilti-objective Optimization Using Non-dominated Sorting in Genetic Algorithms", *Evolutionary Computation*, Vol. 2, No. 3, pp. 221-248.

Taillard, E. D., 1995, "Comparison of Iterative Searches for the Quadratic Assignment Problem", *Location Science*, Vol. 3, No. 2, pp. 87-105.

Tate, D. M., and A. E. Smith, 1995, "A Genetic Approach to the Quadratic Assignment Problem", *Computers & Operations Research*, Vol. 22, No. 1, pp. 73-83.

Zhang, H., C. Beltran-Royo, and L. Ma, 2013, "Solving the Quadratic Assignment Problem by Means of General Purpose Mixed Integer Linear Programming Solvers", *Annals of Operations Research*, Vol. 207, No. 1, pp. 261-278.

Zitzler, E., K. Deb, and L. Thiele, 2000, "Comparison of Multi-objective Evolutionary Algorithms: Empirical Results", *Evolutionary Computation*, Vol. 8, No. 2, pp. 173-195.

Zitzler, E., M. Laumanns, and S. Bleuler, 2004, "A Tutorial on Evolutionary Multi-objective Optimization", *Metaheuristics for multiobjective optimisation*, Springer, New York.