12-2010

AN INVESTIGATION OF METAHEURISTICS USING PATH- RELINKING ON THE QUADRATIC ASSIGNMENT PROBLEM

Thashika Rupasinghe
Clemson University, rthashi@clemson.edu

AN INVESTIGATION OF METAHEURISTICS USING PATH- RELINKING ON THE QUADRATIC ASSIGNMENT PROBLEM

A Dissertation
Presented to the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Industrial Engineering

---

by
Thashika D. Rupasinghe
December 2010

Submitted to:
Dr. Mary. E. Kurz, Committee Chair
Dr. William. G. Ferrell
Dr. Scott J. Mason
Dr. Maria E. Mayorga
Dr. Kevin M. Taaffe

ABSTRACT

The Quadratic Assignment Problem (QAP) is a widely researched, yet complex, combinatorial optimization problem that is applicable in modeling many real-world problems. Specifically, many optimization problems are formulated as QAPs. To resolve QAPs, the recent trends have been to use metaheuristics rather than exact or heuristic methods, and many researchers have found that the use of hybrid metaheuristics is actually more effective. A newly proposed hybrid metaheuristic is path relinking (PR), which is used to generate solutions by combining two or more reference solutions. In this dissertation, we investigated these diversification and intensification mechanisms using QAP. To satisfy the extensive demands of the computational resources, we utilized a High Throughput Computing (HTC) environment and test cases from the QAPLIB (QAP test case repository).

This dissertation consists of three integrated studies that are built upon each other. The first phase explores the effects of the parameter tuning, metaheuristic design, and representation schemes (random keys and permutation solution encoding procedures) of two path-based metaheuristics (Tabu Search and Simulated Annealing) and two population-based metaheuristics (Genetic Algorithms and Artificial Immune Algorithms) using QAP as a testbed.

In the second phase of the study, we examined eight tuned metaheuristics representing two representation schemes using problem characteristics. We use problem size, flow and distance dominance measures, sparsity (number of zero entries in the matrices), and the coefficient of correlation measures of the matrices to build search trajectories.

The third phase of the dissertation focuses on intensification and diversification mechanisms using path-relinking (PR) procedures (the two variants of position-based path relinking) to enhance the performance of path-based and population-based metaheuristics. The current research in this field has explored the unusual effectiveness of PR algorithms in variety of applications and has emphasized the significance of future research incorporating more sophisticated strategies and frameworks. In addition to addressing these issues, we also examined the effects of solution representations on PR augmentation.

For future research, we propose metaheuristic studies using fitness landscape analysis to investigate particular metaheuristics' fitness landscapes and evolution through parameter tuning, solution representation, and PR augmentation.

The main research contributions of this dissertation are to widen the knowledge domains of metaheuristic design, representation schemes, parameter tuning, PR mechanism viability, and search trajectory analysis of the fitness landscape using QAPs.

DEDICATION

This dissertation is dedicated to my parents who gave me life, courage, and education to make me who I am today…

To my loving husband Dil, who has always believed in me, supported me, and was beside me to share all my ups and downs. This wasn't possible if you weren't there...

To our baby who is due November 5, 2010, for her patience and love that kept me going through long hours of work…

ACKNOWLEDGMENT

Pursuing a doctoral degree requires a great deal of effort, patience, and time. My doctoral education not only gave me technical knowledge but also the emotional stability to stay strong when everything else goes wrong! First and foremost, my heartfelt gratitude goes to my advisor, **Dr. Mary Beth Kurz,** who was not only my dissertation advisor but a great friend and mentor. She was always there to support me with intellectual advice as well as the emotional support. The completion of this dissertation would not have been possible without her continued support and guidance. Thank you, Dr. Kurz, for being there for me!

I would like to extend my sincere appreciation to my dissertation committee members, **Dr. Ferrell**, **Dr. Mason**, **Dr. Mayorga**, and **Dr. Taaffe,** for their time, valuable suggestions, and feedback. I would like to thank **Eddie Duffy** (Condor support team at CCIT) for his immense support to run each experiment on Condor. I would also like to offer my gratitude to **Ali Ferguson** at Purple Ink Editing for editing my dissertation and helping me understand the meaning of writing a good dissertation.

I would also like to offer my deepest thanks to **Dr. Anand Gramopadhye**, the department chair of Clemson IE, for his continued support as I completed my dissertation. I am also grateful to him for the opportunities I had while being a part of his research team. Thank you Dr. Gramopadhye!

Last but not the least, I want to thank all of my **friends at Clemson IE (**Kiran Chahar, Esengul Tayfur, Selina Begum, Deepak Vembar, Sajay Sadasivan, Melissa Zelaya, Melissa Dorlette Paul, Kapil Madathil, and Indraneel Dhabade), the **staff** (Saundra Holland, Beverly Robinson, and Martin Clark**)**, and the wonderful **student community** for their support and for being there to remind me that life is beautiful!

Thank you everyone…

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

1. INTRODUCTION

1.1 Research Motivation

This dissertation is motivated by the extensive development of metaheuristics in the combinatorial optimization world and the field's inability to point out why a particular metaheuristic is superior to another. What characteristics enable one method to outperform another? Are these differences due to the methods' varying philosophical differences, problem characteristics, or the detailed metaheuristic design that extends the base implementation? In order to investigate these research questions, we utilize the Quadratic Assignment Problem (QAP), a very well known hard combinatorial optimization problem, as the testbed and investigate a wide variety of metaheuristic-design approaches. Specific problem instances are taken from the QAPLIB problem repository.

1.2 Research Contributions

This dissertation examines the effects of parameter tuning, solution representation, problem characteristics, and path relinking augmentation on four widely applied metaheuristics (Tabu Search, Simulated Annealing, Genetic, and Artificial Immune Algorithms) using the Quadratic Assignment Problem (QAP) as a testbed. The salient features of this study are summarized below:

- **Contribution to the QAP Knowledge Domain**: QAP is one of the most mysteriously difficult combinatorial optimization problems with immense

practical importance. Using QAP as a carrier, we investigate how metaheuristic design factors affect a particular algorithm's performance. Using the results obtained, we classify selected problems from the QAPLIB as trivial, moderately difficult, and hard problem instances. This enables future researchers to investigate these problem characteristics in more detail to learn more about designing effective metaheuristics.

- **Large Numbers of Test Cases:** We utilize 130 test cases from the QAPLIB with problem sizes ranging 12-128 facilities/locations. Many of these problems are based on real-world data, which is helpful for practitioners when evaluating the true performance of a particular metaheuristic. Since the cluster of studies presented here uses a large number of test cases, this generates robust and replicable findings as opposed to many of the studies mentioned in the literature, which have used only a small number of test cases.
- **The Effects of Parameter Tuning:** An extensive parameter tuning procedure was carried out for the four selected metaheuristics to investigate the effects of parameter tuning on performance. Using a full factorial design framework, we identified metaheuristic design factors for each method and replicated each treatment in a high throughput computing environment.
- **The Effects of Solution Representation:** We investigated how the performance of a particular metaheuristic is affected by the solution representation scheme. We used random keys and permutation representation schemes and analyzed how each metaheuristic method is affected by each representation scheme.

- **The Effects of Path Relinking Augmentation:** We implemented position-based path relinking (POS_PR) mechanisms for the two representation schemes, and we completed comparison studies for each representation scheme. We found that PR can be considered an efficient diversification mechanism and that metaheuristic designers should pay careful attention to representation schemes as well as the philosophical differences of a particular metaheuristic.
- **Performance Measures:** We used the number of fitness evaluations as an absolute measure and run time as a relative measure with the computation of average loss function values (average deviation from the best known solution). We use 50 replications to compute the average loss function values for each metaheuristic.
- **Problem Characteristics:** We investigated the performance of each tuned path-based and population-based metaheuristic using measures pertaining to their problem characteristics. We used problem size, flow and distance dominance measures, sparsity (number of zero entries in the matrices), and the coefficient of correlation measures of the matrices to build search trajectories. In order to compare and contrast each population-based method with the path-based methods, we used Classification and Regression Tree (CART) tools.
- **Individual Search Trajectories:** The tuned algorithms were analyzed using individual search trajectories when stagnation occurs in the life cycle. The search trajectories were also monitored for each metaheuristic and for each representation scheme.

- **High Throughput Computing (HTC):** The full factorial design for the four algorithms with the large set of input files requiring 50 replications for each run took 1,022,852 computation hours. This large-scale computational study utilized an HTC environment, or else the experiment would have taken 116.6 years of regular computing. The other experimental runs (problem characteristics and PR augmentation) also required extensive computational resources. As such, this dissertation was enabled by the HTC's efficient computation architecture.
- **Parametric and Nonparametric Statistical Analysis:** This extensive computational study utilized parametric and nonparametric (Friedman's test and Mood Median test) statistical analysis tools and techniques using R®, Mini Tab®, JMP® statistical software to draw conclusions.

From this dissertation, we draw conclusions related to each of the categories listed above, which are explained in detail in each chapter of the dissertation.

1.3 Dissertation Overview

This dissertation consists of six chapters including the introduction and concluding remarks. Figure 1.1 depicts the layout of each chapter and how the chapters build on each other.

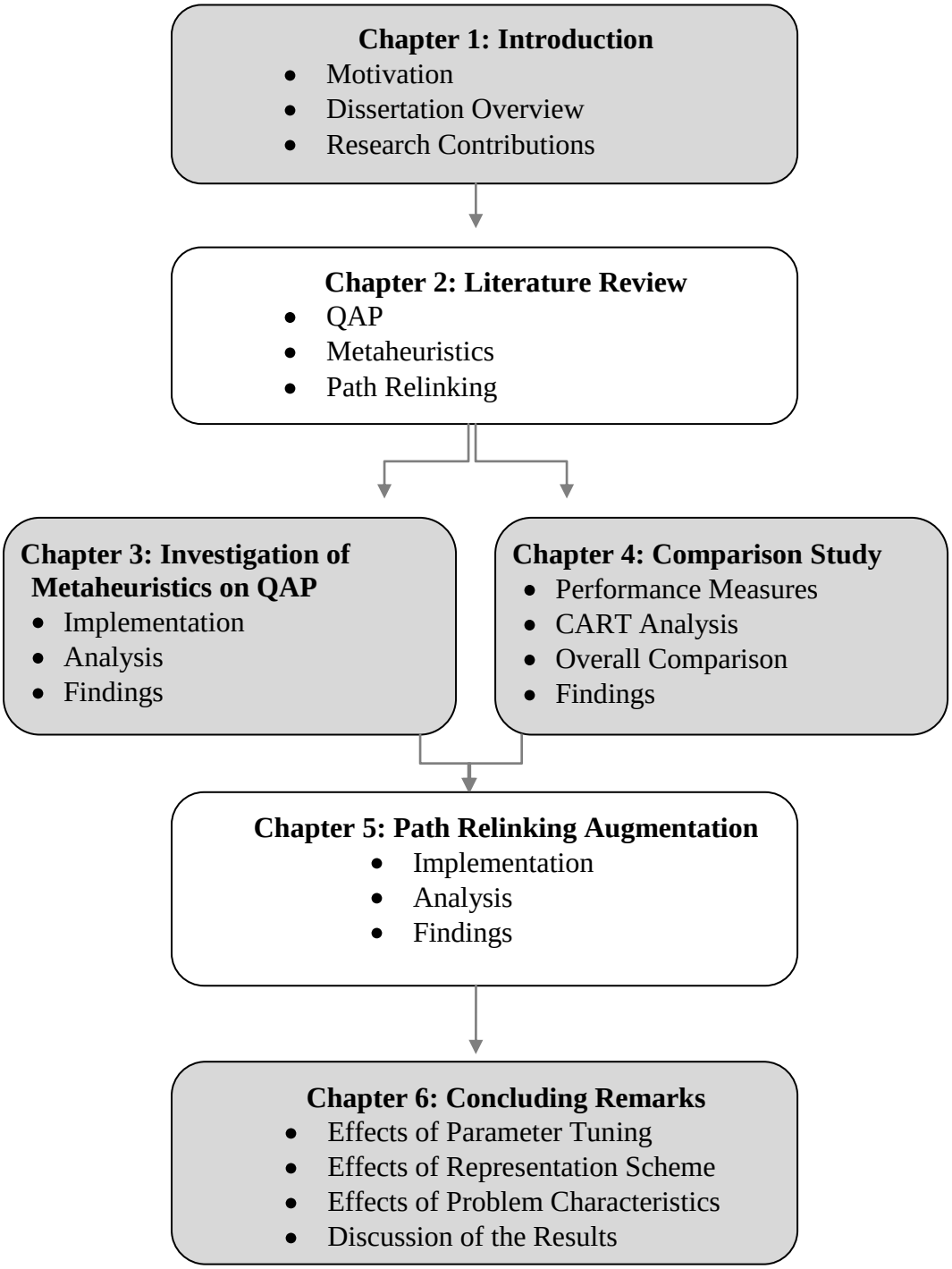


Figure 1.1: Overview of the Dissertation

CHAPTER TWO

2. LITERATURE REVIEW

2.1. Introduction

The Quadratic Assignment Problem (QAP) is a widely researched combinatorial optimization problem, due to its complexity as well as its practical and theoretical importance. In recent years, researchers have shown promising results in solving complex QAPs with metaheuristics, specifically focusing on the development of efficient algorithms. Widely applied metaheuristics, such as Genetic Algorithms, Tabu Search, and Simulated Annealing, have been applied to approximate solutions for many combinatorial problems for which optimal solutions from exact mathematical models are unavailable.

The second chapter of this dissertation begins with a thorough exploration of the previous research conducted in the field of QAPs, metaheuristics and Path-Relinking. The literature review is organized as follows. In the first sub-section, the Quadratic Assignment Problem and related mathematical models are presented. The resolution approaches proposed for QAPs are presented with a brief introduction to the exact mathematical models and approximation algorithms. The test cases and the lower bounds are discussed following the introduction to metaheuristics. This section elaborates on the evolution of metaheuristics, the hybridization of metaheuristics, and the inception of Path-Relinking procedures. In the later sub-sections of the chapter, the computational framework and notations are introduced. The chapter concludes by

highlighting the significance of the dissertation and by laying the foundation for chapter 3 to investigate metaheuristics for QAPs.

2.1.1. The Quadratic Assignment Problem (QAP)

The Quadratic Assignment Problem (QAP) was first introduced as a mathematical model by Koopmans and Beckman in 1957 [54]. Many practical problems can be modeled as a QAP, including production line scheduling, assignment of gates to airplanes in airports, backboard wiring problems in electronics, campus and hospital layouts, typewriter keyboard designs, turbine runner balancing problems, processor-to-processor assignments in a distributed processing environment, and many others. On account of its diverse applications, its theoretical importance, and its overall complexity, QAP has been studied by many researchers around the world [58].

2.1.2. Mathematical Formulations

The mathematical formulation of the QAP takes different forms, including integer linear programming (IP) formulation, mixed integer linear programming formulation (MIP), formulation by permutations, tree formulation, and graph formulation. Generally, the IP format is used to formulate the QAP, which is usually described within the context of a facility location problem [58]. The intent of such QAPs is to assign facilities to locations in such a way that each facility is located in exactly one location and vice-versa. These decisions are represented by the decision variables, x_{ik}, which take on the value 1 when the facility *i* is located in location *k* and 0 otherwise. There are two data matrices associated with the problem: the distances, d_{kp}, between locations *k* and *p* and the demand flows, f_{ij}, between facilities *i* and *j*. The

facilities are assigned such that the sums of all possible distance-flow products are minimized [54]. The following is a standard integer programming formulation:

$$\min \quad \sum_{i,j=1}^{n} \sum_{k,p=1}^{n} f_{ij} d_{kp} x_{ik} x_{jp} \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ij} = 1 \quad 1 \le j \le n, \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad 1 \le i \le n, \tag{3}$$

$$x_{ij} \in \{0,1\} \quad 1 \le i, j \le n, \tag{4}$$

Another complementary version to the general form was proposed by Lawler in 1963 by substituting the **A**, **B,** and **C** matrices with **C**= [c_{ijkp}] to represent the total cost [58]. The Lawler formulation is as follows:

$$\min \quad \sum_{i,j=1}^{n} \sum_{k,p=1}^{n} c_{ijkp} x_{ik} x_{jp} \tag{5}$$

s.t. (2), (3) and (4)

The above discussed formulations only differ in how they are written, not in their intent. In a more general form, we can identify a QAP instance of order *n* by three matrices, **A**= [f_{ij}], **B**= [d_{kp}], and **C** = [c_{ik}], where the first two matrices define the flows and distances between the facilities and their locations, respectively. Matrix **C** is the allocation cost of facilities to locations, resulting in the following formulation:

$$\min \quad \sum_{i,j=1}^{n} \sum_{k,p=1}^{n} f_{ij} d_{kp} x_{ik} x_{jp} + \sum_{\mathrm{i,k=1}}^{\mathrm{n}} c_{ik} x_{ik} \tag{6}$$

s.t. (2), (3) and (4)

The term *quadratic* stems from the formulation of the QAP as an integer optimization problem with a quadratic objective function stemming from the $x_{ik}x_{jp}$ terms, as in equation (1).

The QAP is a NP-hard optimization problem; Sahni and Gonzales [79] show that unless P=NP, it is not possible to find an *f*-approximation algorithm for a constant *f* for this problem. Even with rapid growth in computational resources, solving relatively large instances (>20) of QAPs is considered intractable. Due to its high computation complexity, QAP was chosen as the first major test application for the GRIBB project (Great International Branch-and-Bound search), which intends to develop a software library for solving a large class of parallel search problems using a number of computers around the globe via the internet [58].

Many real world QAPs have problem sizes of more than 20 facilities or locations to be assigned; hence, the use of heuristic methods has been common in recent years [5]. Before the 1980s, most of the proposed heuristic methods were problem and domain specific. However, recently, more research studies have begun to look at classes of heuristics, widely known as metaheuristics, as being applicable for more generalized contexts [11]. In the absence of optimal solutions, these methods provide fairly good solutions within reasonable timeframes [2].

2.1.3. Variants of QAPs

As the knowledge base has grown in the QAP literature, several augmentations have been proposed. One such variant to the original QAP includes Steinberg's [81] Quadratic Bottleneck Problem (QBAP), which is related to the backboard wiring problem. In this problem *n* components are allocated to individual

locations with the objective to minimize the maximum length of wire needed to connect two given components. Other variants include Pierskalla's [74] Quadratic three-dimensional Assignment Problem (Q3AP) for data transmission system design and Hansen and Lih's [43] Quadratic Semi-Assignment Problem (QSAP). These problems are extensions to the classical QAP and have shown a wide spectrum of applications.

Other combinatorial problems that can be formulated as QAPs include the Graph Partitioning Problem (GPP) [35], the Maximum Clique Problem (MCP) [7], the famous Travelling Salesman Problem (TSP) [29], and the Packing Problem in graphs [58]. These problems can be formulated as QAPs with use of relevant data matrices. Understanding the relationship between these problems and QAP allows researchers to gain insights into QAPs behavior, complexity, and possible resolution.

2.1.4. Applying QAPs

From the vast body of research published on applying QAP, the following summary was created to highlight the importance of investigating this problem domain (Table 2.1).

Table 2.1: Some Applications of QAP from the Literature, Adapted from [58]

Research Studies	**Description of the QAP Application**
Steinburg (1961)	Backboard wiring problems
Heffley (1972,1982)	Economic problems
Francis and White (1974)	Decision framework for assigning new facilities
Geoffrion and Graves (1976)	Scheduling problems
Pollatscheck *et al.* (1976)	Typewriter keyboards and control panels
Krarup and Pruzan (1978)	Archeology
Hurbert (1987)	Statistical analysis
Forsberg *et al.* (1994)	Analysis of reaction chemistry
Dickey and Hopkins (1972)	Assignment of buildings at a university
Elshafei (1977)	Hospital planning
Bos (1993)	Forest parks

Benjaafar (2002)	Minimizing work-in-progress (WIP)
Ben-David and Malah (2005)	Error control in communication via index assignment problems
Wess and Zeitlhofer (2004)	Memory layout optimization in signal processors

2.1.5. Solution Approaches

Since its inception, many researchers have investigated possible methods to solve QAP, to obtain both optimal and near optimal solutions. With the absence of tractable exact mathematical models, heuristics have become increasingly preferred in recent years [6] and many researchers have shown the efficiency of these methods to generate fairly good solutions within reasonable timeframes.

The following two sub-sections summarize the two broad categories of resolution approaches: exact mathematical approaches and approximation algorithms.

2.1.6. Exact Mathematical Approaches

Using the classical mathematical modeling techniques, exact mathematical models generate optimal solutions for a given optimization problem. In the case of QAP, the most popular exact mathematical solution methods include branch-and-bound techniques, cutting planes, and dynamic programming. Table 2.2 presents a summary of these methods [58].

Table 2.2: Summary of the Solution Methods for QAP Adapted from [58]

Research Studies	**Solution Method**
Gavett & Plyter (1966), Nugent *et al.*(1968), Graves & Whinston (1970), Pierce & Crowston (1971), Burkard & Stratman (1978), Bazaraa & Elshafei (1979), Mirchandani & Obata (1979), Roucairol (1979), Burkard & Derigs (1980), Edwards (1980), Bazaraa & Kirca (1983), Kaku & Thompson (1986), Pardalos & Crouse (1989), Burkard (1991),	Branch-and-bound

Laursen (1993), Mans *et al.* (1995), Bozer and Suk-Chul (1996), Pardalos *et al.* (1997), Brüngger *et al.* (1998), Ball *et al.* (1998), Spiliopoulos & Sofianopoulou (1998), Brixius & Anstreicher (2001), Hahn *et al.* (2001a,b)	
Roucairol (1987), Pardalos & Crouse (1989), Mautor & Roucairol (1994a), Brüngger *et al.* (1997), Clausen & Perregaard (1997)	Branch-and-bound with parallel implementation
Christofides & Benavent (1989),Urban (1998)	Dynamic programming
Bazaraa & Sherali (1980), Kaufman & Broeckx (1978), Bazaraa & Sherali (1980, 1982) , Burkard & Bonniger (1983)	Cutting planes
Miranda *et al.* (2005)	Benders decomposition
Padberg & Rinaldi (1991)	Branch-and-cut technique
Jünger & Kaibel (2000, 2001a,b), Padberg & Rijal (1996), Kaibel (1998), Blanchard *et al.* (2003)	Investigation of properties of polytopes

Even with the rapid development of computational recourses, an exact solution for any QAP instance of size n=20 was not found until the mid-1990s when Mautor and Roucairol [63] presented an exact optimal solution to the *nug16* QAP [18] for the first time. Table 2.3 depicts a brief history of the exact solutions of several problem instances of QAPLIB [47].

Table 2.3: History of Solutions for Some Problem Instances of QAPLIB

Problem Instance	**Research Study**
Nug16	Mautor & Roucairol (1994)
Nug20	Clausen &Perregaard (1997) (Branch-and-bound technique and 960 min of computation and 16 processors)
Nug25	Marzetta & Brüngger (1999) (Dynamic programming and parallel implementation with 64 and 128 processors and 30 days of computation)
Nug25	Anstreicher & Brixius (2001) (Convex quadratic programming relaxation within a branch-and-bound

	algorithm with 6.7 wall-clock time)
Kra30a	Hahn & Krarup (2001) (99 days of computational time on a sequential workstation)
Ste36b and Ste36c	Nystrom (1999) (Distributed programming environment with 22 processors. The solution took approximately 60 days for Ste36b and 200 days for Ste36c of computation time)
Ng30, Nug28, Nug30, Kra30b, and Tho30	Anstreicher *et al.* (2002) (7 days to complete on a computational grid with an average of 650 computers simultaneously processing)

2.1.7. Approximation Algorithms

For the last three decades, the use of heuristics or approximation algorithms has increased drastically [76]. Various sub-categories of metaheuristics have delivered promising results in solving complex QAPs in recent research [6]. Table 2.4 depicts a brief summary of the heuristics applicable for solving QAPs.

Table 2.4: Summary of the Heuristics Applicable for QAP Adapted from [58]

Research Studies	**Heuristics/ Metaheuristics**
Gilmore (1962), Armour & Buffa (1963), Buffa *et al.* (1964), Sarker *et al.* (1995, 1998), Tansel & Bilen (1998), Burkard (1991), Arkin *et al.*(2001), Gutin and Yeo (2002),Yu &Sarker (2003)	Constructive methods
Burkard & Bonniger, (1983), West (1983), Nissen & Paul (1995)	Enumerative methods
Heider (1973), Mirchandani & Obata(1979), Bruijs (1984), Pardalos *et al.* (1993), Burkard & Cela (1995), Li & Smith (1995), Anderson (1996), Talbi *et al.*(1998a), Deineko &Woeginger (2000) ,Misevicius (2000), Mills *et al.*(2003)	Improvement methods
Burkard & Rendl(1984),Wilhelm & Ward (1987), Connolly (1990), Abreu *et al.* (1999), Bos (1993), Yip & Pao (1994), Burkard and C`ela (1995), Peng *et al.* (1996), Tian *et al.* (1996, 1999), Mavridou & Pardalos (1997), Chiang & Chiang (1998), Misevicius (2000b, 2003c), Tsuchiya *et al.* (2001), Siu & Chang (2002), Baykasoglu (2004).	Simulated Annealing
Bui & Moon (1994), Tate & Smith (1995), Mavridou & Pardalos (1997), Kochhar *et al.* (1998), Tavakkoli-Moghaddain & Shayan (1998), Gong *et al.* (1999), Drezner & Marcoulides (2003), El-Baz (2004), Wang & Okazaki (2005), Drezner (2005a)	Genetic Algorithms

Cung *et al.* (1997)	Scatter Search
Maniezzo & Colorni (1995, 1999), Colorni *et al.* (1996), Dorigo *et al.* (1996), Gambardella *et al.* (1999), Stu¨tzle & Dorigo (1999), Stu¨ tzle & Holger (2000), Talbi *et al.* (2001), Middendorf *et al.* (2002), Solimanpur *et al.* (2004), Randall (2004),Ying & Liao (2004), Acan (2005)	Ant colony Optimization
Skorin-Kapov (1990, 1994),Taillard (1991), Bland & Dawson (1991), Rogger *et al.* (1992), Chakrapani & Skorin-Kapov (1993), Misevicius (2003a, 2005), Drezner (2005b)	Tabu Search
Li *et al.* (1994b), Feo & Resende (1995), Resende *et al.* (1996), Fleurent & Glover (1999), Ahuja *et al.* (2000), Pitsoulis *et al.* (2001), Rangel *et al.* (2000), Oliveira *et al.* (2004)	Greedy Randomized Adaptive Search Procedure (GRASP)

2.1.8. Lower Bounds

Several lower bounds have been proposed in the literature. These methods are the building blocks for the branch-and-bound methods and are essential for heuristics to evaluate the solution quality. The lower bound proposed by Gilmore in 1962 and then by Lawler in 1963, which is now known as the Gilmore and Lawler bound (GLB), is widely applied to determine lower bound for QAPs due to its effectiveness in small problem instances [58]. However for large test cases, this method proves to have inferior results. Another category of bounds was proposed for Mixed Integer Linear Programming (MILP) formulations of QAPs. These lower bounds are based on GLB reformulations, interior point methods, variance reduction bounds, graph formulations, spectral bounds, semi-definite programming, and reformulation-linearization bounds [5], [17], [58].

2.1.9. Input Data and Characteristics

The characteristics of the data matrices of the QAP (flow and the distance matrices) play an important role in obtaining a good solution for a given QAP [86].

For a single objective QAP, one flow and one distance matrix are used. In order to differentiate categories of problem instances, researchers have proposed flow dominance (fd) and distance dominance (dd) measures using the coefficient of variations in the data matrices [64]. The following formulae have been used for computing the flow and distance dominance measures, where n is the size of the problem and b_{ij} is the i^{th} and j^{th} value of the flow or distance matrices [88].

$$f(\text{Distance or Flow}) = 100\frac{\sigma}{\mu} \tag{7}$$

where

$$\mu = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij} \quad \text{and} \quad \sigma = \sqrt{\frac{1}{n^2-1}\sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij} - \mu^2} \tag{8}$$

Using these measures one can describe the characteristics of given data matrices, where high fd and dd measures indicate that the majority of the data is clustered among a few facilities. If the entries are equally sized, fd is low; however, for irregular instances fd values are high. The fd values are also an indication of how local optimum solutions are scattered. For example, high fd values suggest that there are few small local optima and few large local optima, which create difficulties for local search algorithms. Randomly generated problem instances using a uniform distribution for the flows and distances show low fd and dd values. Real-world problems and non-uniformly generated random instances show high fd and dd values.

Another important measure is the sparsity of the data matrices. This is associated with the number of zero elements in the flow and distance matrices. Problems with sparse data matrices are less likely to realize improvements through

pair-wise interchanges, as the data matrices are dominated by zero elements and interchanges cannot reduce the total costs drastically [31].

A repository of problem instances called QAPLIB, which was compiled by Burkard *et al.*[18], presents more than 135 input files related to QAP. These instances consist of symmetric, asymmetric, and rectangular data from pseudorandom number generation as well as from actual data collected from hospital layouts, backboard wiring problems, and many other real-world applications. These problems have either been optimally solved by exact mathematical methods or have been approximated by metaheuristics. The problem sizes range from 12 to 256, with 50 problem instances up to size 20 being optimally solved to date [77].

2.2. Metaheuristics

The term *metaheuristics* was first proposed by Glover [37] to refer to a broad class of algorithmic concepts used for optimization and problem solving. According to Voß *et al.* [88] "a metaheuristic is an iterative master process that guides and modifies the operations of subordinate heuristics to efficiently produce high quality solutions."

Some of the metaheuristic techniques are based on natural process metaphors and some are based on theoretical and experimental considerations [76]. The metaheuristics that are inspired by nature include Genetic Algorithms (GA) (the most popular process) [3], [13], [44] Simulated Annealing (SA) [12], [50], [61] Scatter Search (SS) [21], [42] and Ant Colony Optimization (ACO) [36]. The other category of metaheuristics that has been applied to QAP settings includes Tabu Search (TS)

[12], [37], [45], [46] the Greedy Randomized Adaptive Search Procedure (GRASP) [34], and Variable Neighborhood Search (VNS).

2.2.1. The Evolution of Metaheuristics

The broad class of metaheuristics can be further categorized into sub-groups based on their philosophical differences, which are apart from their differences in the inspirational metaphors. Generally the nature-inspired metaheuristics to QAPs incorporate randomness. Based on their search mechanisms, these broad categories can be further broken down into population-based and path-based subsets [11], [75], [76]. Population-based metaheuristics consist of Genetic Algorithms, Ant Colonies, and Scatter Search algorithms, which consider ways to coalesce and extend the elements of the solutions that already exist. On the other hand, path-based algorithms, such as Tabu Search and Simulated Annealing, incorporate strategies to transform a single solution.

Research conducted in the field of metaheuristics includes two paradigms. The first paradigm takes several known metaheuristics as a subset of several general methodologies to explore new problems or extend previous problem instances with less emphasis on metaheuristic design. The other paradigm thoroughly redesigns and tunes one design to evaluate its impact on one or more test problems. By incorporating these methodologies, researchers attempt to contribute to the field of metaheuristics by building robust designs that resolve complex combinatorial optimization problems.

2.2.2. The Hybridization of Metaheuristics

Recent trends have focused on hybridizing pure metaheuristic strategies to exploit the resolving power of the designs in terms of solution quality and efficiency. A number of algorithms have been reported that do not completely follow the concepts of a single traditional metaheuristic*;* rather, they combine various algorithmic ideas, sometimes going beyond the boundaries of traditional metaheuristics. These approaches are commonly referred to as hybrid metaheuristics [77], [82], [86], [91]. Researchers' motivation behind hybridizing different algorithmic concepts is usually to obtain better performing systems that exploit and unite the advantages of the pure strategies through synergy.

Hybridization is approached via several forms starting with *what* we hybridize (i.e., which kind of algorithms should be combined). We might combine any of the following:

(a) Different metaheuristic strategies

(b) Metaheuristics with certain algorithms specific for the problem we are considering

(c) Metaheuristics with other more general techniques coming from fields like Operations Research (OR) and Artificial Intelligence (AI).

Then, the level or the strength of the retaining identities should also be also considered. The third property to consider is the order of execution followed by the control strategy, which determines the combinations via integrative (coercive) and collaborative means [75].

Several researchers have investigated the concept of hybridization in the QAP context [58]. In a recent study, Drenzner [30] states that the importance of

hybridization within the QAP context is to enhance the solution quality and the immense opportunities available for future research. Table 2.5 depicts some possible components available for the hybridization of the most commonly known metaheuristics [72].

Table 2.5: Key Components of Popular Metaheuristics

Metaheuristics	**Possible Components for Hybridization**
	OF: Output function **IF**: Input function **IM**: Improvement method **SCM**: Solution combination method
ACO	**OF:** Derivation of new solution candidates by considering a pheromone matrix **SCM:** Implicitly via the pheromone matrix **IF:** Updates of the pheromone matrix
GA	**OF, IF:** Selection techniques **SCM:** Crossover operators **IM:** Mutation operators, repair schemes, decoding functions
PR	**OF, IF:** Selection techniques **SCM:** Crossover operators **IM:** Mutation operators, repair schemes, decoding functions
SS	**IF:** Diversification generation methods, subset generation methods **IM:** Improvement methods **SCM:** Solution combination methods **OF:** Reference set update methods
SA	**IF:** Acceptance criterion, annealing schedule
TS	**IM, IF:** Consideration and maintenance of Tabu list, aspiration criteria

2.3. Path Relinking (PR)

First proposed by Glover [38], path relinking (PR) is a novel instance of traditional evolutionary metaheuristics that stems from Scatter search. PR embodies principles and strategies that have still not been emulated by other evolutionary methods, a fact that proves to be advantageous for solving a variety of complex problems [41].

Similar to other evolutionary metaheuristics, PR operates with a population of solutions, rather than with a single solution at a time, and employs procedures for "combining" these solutions to create new solutions [38]. Two of the most distinguishing features of PR are its alliance with Tabu Search (TS) and its adoption of the principle that a search can benefit by incorporating special forms of adaptive memory. Path-based metaheuristics, such as TS, can utilize PR techniques to explore neighborhoods effectively. This enables researchers to combine solutions of good quality to generate intelligent neighborhood search paths. For population-based metaheuristics the possibilities for utilizing PR are endless. It can be incorporated via crossover and mutation operators, elite reproduction, combining generations and populations of solutions through tunneling, and much more.

Since the inception of the PR concept, researchers have investigated its applicability in different contexts, including vehicle routing, arc routing, financial product design, neural network training, job shop scheduling, crew scheduling, flow shop scheduling, unconstrained optimization, optimization simulation, multi-objective assignment problems, and quadratic assignment problems [41]. These studies have contributed several improved methods for solving a variety of classical problems.

The theory behind the PR concept originates from SS, which combines a certain set of solutions, the *reference set*, to create new solutions. The main mechanism for combining solutions is through a linear combination of two other solutions. In a similar way, convex and non-convex combinations of both original and new reference solutions create more solutions, thereby generating a more complete *reference set* [38][39].

2.3.1. Path Relinking Framework

In the PR framework, from a spatial orientation, the process of generating linear combinations from a reference solution set may be characterized as generating paths between and beyond these solutions, where solutions on such paths also serve as sources for generating additional paths. Such combinations generate paths between and beyond selected solutions in a neighborhood space rather than in a Euclidean space [39].

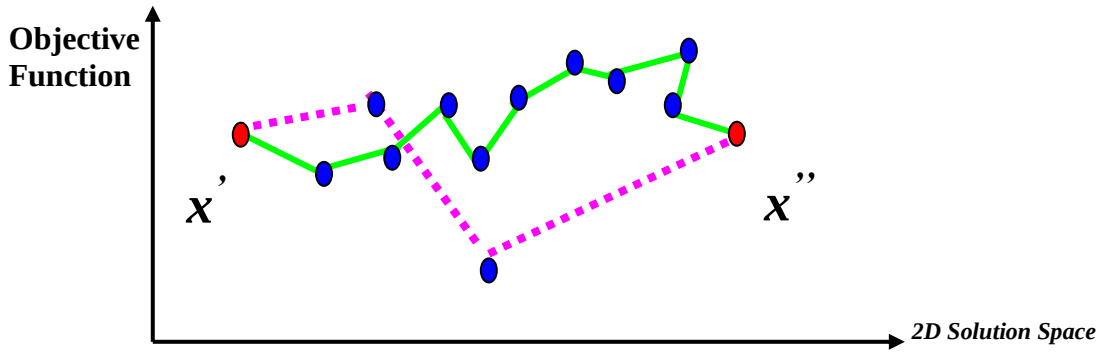


Figure 2.1 Path Relinking: Original path shown by the heavy line and a possible relinked path shown by the dotted line [38]

Figure 2.1 depicts path creation that join two selected solutions x' and x'', which restricts the attention to the part of the path that lies "between" the solutions and produces a solution sequence: $x' = x(1), x(2), \ldots, x(r) = x''$. To reduce the number of options to be considered, the solution $x\ (i + 1)$ may be created from $x\ (i)$ at each step by choosing a move that leaves a reduced number of moves remaining to reach x. This policy permits a significant number of alternative choices for generating the next solution for each step.

2.3.2. PR and Scatter Search Strategies

PR strategies have demonstrated promising results in assignment problems [4], [92]. Exploring such problems, Alfandariet *et al.* [4] and Yagiura *et al.* [92]

investigated the Generalized Assignment Problem (GAP) using PR concepts. In the literature, problem instances of Generalized Assignment Problem (GAP) can be categorized into several types (A through E) depending on the problem characteristics and type D and E are known to be very difficult. PR strategies were able to generate solutions that are not only highly effective in general but are also especially effective for solving types D and E instances [92]. These researchers also suggest that adopting more sophisticated strategic rules could result in better performance and that pursuing possibilities related to these rules is an important direction for further research. Furthermore, the PR approach is quite powerful even in its simplest form and is not sensitive to slight changes in the rules and parameters involved in its framework [42]. Therefore, exploration of such mechanisms in the QAP context supports the development of more robust metaheuristics.

Scatter search has been applied in the context of QAP and has yielded highly effective solutions, even though many variants of SS have not even been fully exploited [39]. Recently, sequential and parallel PR-based TS algorithms have been proposed for QAP, and the computational results have demonstrated highly attractive outcomes [45], [46]. James *et al.* [45] investigated QAP in the context of PR, and the authors encourage additional follow-up studies that examine more complete and advanced forms of PR that make use of more sophisticated processes for managing the *reference set* and for creating combination solutions.

2.3.3. Recent Advances in QAP domains

PR has been applied to QAP with Greedy Randomized Adaptive Procedures (GRASP) and has shown to significantly improve the time to find a target solution [71].

As a form of path-crossover in GAs, PR strategies have been applied to the QAP context and have shown very promising results by finding the best known solutions to all the problems tested [3]. Ahuja *et al.* [3] developed a greedy Genetic Algorithm was proposed and tested on test cases of QAPLIB with problem sizes less than 100. The authors suggest that by incorporating more efficient and advanced PR mechanisms, better solutions can be obtained.

In a subsequent study, a more rigorous analysis was carried out with different crossover operators [69]. Misevičius and Kilda [67] used ten different crossover operators, including swap path crossover (SPX), which is related to PR. The study used nine random and ten real-world test cases of QAPLIB for which SPX generated very effective solutions.

A recent study addressed the important issue of search bias of crossover operators using QAP [86]. Thierens [86] compared random PR and greedy PR crossover operators with uniform permutation crossovers (UPX) using five problem instances [65] and PR-based crossover operators generated solutions effectively and efficiently.

2.4. Computational Resources

In order to solve large and complex QAPs, past research has often utilized parallel processing hardware [5]. Table 2.6 depicts the computation demands of a selected few complex QAP instances. While these computational methods are very promising the capabilities of these supercomputers are emerging and thus create issues related to costs and availability. An alternative for this method is the High-Throughput Computing (HTC) environment, which utilizes multiple machines that are interfaced with some form of a communication network. The terms “metacomputing” and “grid computing” refer to a very large scale distributed computation associated with machines which are geographically dispersed [9].

Table 2.6: Recently-solved large QAPs [9]

Problem	Bound	Platform	CPU days
kra30a	Dual -LP	Serial	99
kra30b/32	Quadratic programming bounds (QPB)	Distributed	1527/5536
nug27/28/30	Quadratic programming bounds (QPB)	Distributed	113/722/3999
ste36a	Gilmore-Lawler bound (GLB)	Serial	18
ste36b/c	Gilmore-Lawler bound (GLB)	Distributed	60/200
tai25a	Dual -LP	Serial	394
tho30	Quadratic programming bounds(QPB)	Distributed	8997

Condor is a specialized high-throughput computing system for compute-intensive tasks [57]. Condor can be used to build grid-style computing environments that cross administrative boundaries. Its "flocking" technology allows multiple Condor computer installations to work together and incorporates many of the emerging grid-based computing methodologies and protocols. While providing functionality similar to that of a more traditional batch-queuing system, Condor's

novel architecture allows it to succeed in areas where traditional scheduling systems fail. Condor can be used to manage a cluster of dedicated computer nodes. In addition, unique mechanisms enable Condor to effectively harness wasted CPU power from otherwise idle desktop workstations [78].

2.5. Notation and Terminology

Table 2.7 presents a summary of the notations used throughout this dissertation.

Table 2.7 Summary of Notations

Notation	***Description***
n	The size of an instance of the QAP
A	The flow matrix
B	The distance matrix
dd	Distance dominance
D&I	Diversification and Intensification mechanisms
fd	Flow dominance
GA	Genetic Algorithm
IA	Artificial immune Algorithm
PR	Path-Relinking
QAP (A,B)	An instance of the QAP with flow matrix A and distance matrix B
SA	Simulated Annealing
SS	Scatter Search
TS	Tabu Search

2.6. Significance of the Dissertation

This dissertation contributes to three diverse but integrated categories of the metaheuristic domain:

i. Parameter tuning processes, representation scheme, and metaheuristic design

ii. Investigation of performance of metaheuristics using detailed problem characteristics

iii. Diversification and Intensification mechanisms for intelligent searches via PR mechanisms

Using three integrated studies, layers of the above mentioned metaheuristic categories will be examined. Findings from each proceeding study will be integrated into the succeeding study to continuously improve the proposed algorithms.

The overall findings of this dissertation will be applicable for general metaheuristic development, large-scale computational comparisons, extensive parameter tuning procedures, broad metaheuristic design rules, combinatorial optimization, and unified framework for QAP-related metaheuristics.

CHAPTER THREE

3. INVESTIGATION OF METAHEURISTICS ON QAP

3.1. Introduction

The Quadratic Assignment Problem (QAP) is widely researched due to its complexity and its practical and theoretical importance. In recent years, metaheuristics have shown promising results in solving complex QAPs, with researchers specifically focusing on the development of efficient algorithms. However, less emphasis has been given to exploring why certain methods outperform others. This study utilizes two path-based metaheuristics (Tabu Search and Simulated Annealing) and population-based metaheuristics (Genetic Algorithms and Artificial Immune Algorithms) that are widely applied in solving QAPs and investigates the impact of parameter tuning, solution representation, and metaheuristic design on performance. An extensive parameter-tuning process is carried out in a high throughput computing environment with 130 test cases from the QAPLIB (QAP test cases). The comparative studies are followed by nonparametric and parametric statistical analyses, and conclusions are drawn with indications for possible future research.

3.2. Metaheuristic Comparison Studies

The earliest comparison of several metaheuristics dates back to 1993 [80] in which Tabu Search (TS), Simulated Annealing (SA), Genetic Algorithms (GA), and several other heuristics were compared using the hydraulic turbine runner-balancing problem (a special case of QAP). This study mainly evaluated the applicability of

metaheuristics when real data was used. Battiti and Tecchiolli [12] utilized SA and Reactive TS and compared their performance based on run-time and number-of-fitness-function evaluations using five problem instances (size < 50) rather than design parameters or generic characteristics. Using eight metaheuristics implemented on a unified computer system called ALGODESK, Maniezzo *et al.* [61] compared performance using eight problem instances. Merz and Freisleben [64] used GA, TS, and memetic algorithms (MA) for QAP, focusing on comparing the performance of an improved MA to others using eleven problems taken from the QAPLIB [18].

These studies either investigated parameter tuning or conducted a comparative analysis using multiple metaheuristics to draw conclusions regarding the superiority of a particular algorithm(s), specifically analyzing those with fewer problem instances. However, less emphasis was given to evaluating the effects of parameter tuning, solution representation, metaheuristics design, and the underlying characteristics of the algorithms using a large set of problems instances. In order to fill this gap in the literature, our study designed and utilizes high throughput computational resources to complete the extensive computational experiments successfully. The research questions/statements addressed in this chapter include the following:

- What is the significance of solution representations (random keys and permutations-solution-encoding procedures) on the performance of each path-based and population-based algorithm?
- What is the significance of parameter tuning on the performance of each path-based and population-based algorithm?

- Categorize the 130 input files from QAPLIB as trivial, moderate, or hard problems.

This chapter is organized as follows: We begin with the implementation of the four algorithms and related sub-sections. Then we move on to discuss the performance measures, experimental results, and finally the conclusions and possible extensions to this research.

3.3. Implementation of the Four Metaheuristics

In this section, a detailed description of the implementation of the four algorithms is presented. For this study, we have implemented the metaheuristics as they were proposed by the original authors without any other modifications with the intention being to study the match between the inherent characteristics and the performance of the algorithms in their original form.

3.3.1. Simulated Annealing Algorithm (SA)

Simulated Annealing was developed based on the analogy of annealing solids to mimic optimization problems [50]. This metaphor is used for modeling optimization problems in which the particles represent the solutions and the level of energy refers to the value of the fitness or the objective function. An abstract description of a generic SA is given in Figure 3.1.

Initialize a candidate solution $\boldsymbol{s}$
Initialize the temperature, cooling schedule, and the number of iterations
While the temperature has not reached the equilibrium,
    $\boldsymbol{s}^{new}$ =local search
        If $\boldsymbol{s}^{new}$ is accepted based on criterion $\exp^{(f_1 - f_0)/\text{TEMPERATURE}}$,
            then move towards $\boldsymbol{s}^{new}$
    Update the temperature and record solutions

Figure 3.1: Implementation of a Generic SA

3.3.2. Tabu Search Algorithm (TS)

Tabu Search (TS) has been widely applied to many optimization problems [37][38] and is based on the underlying metaphor of human memory to track solution trajectories. The efficient search strategies inherent in this method yield remarkably good solutions for hard optimization problems. A generic TS implementation [37] is presented in Figure 3.2.

```
Initialize a candidate solution s
Initialize the tabu list and number of iterations
While number of iterations has not reached the maximum,
    Create a neighborhood s^new =local search(es)
        If s^new is accepted based on tabu list or aspiration criterion,
           then move towards s^new
        Update the tabu list and record solutions
```

Figure 3.2: Implementation of a Generic TS

3.3.3. Genetic Algorithm (GA)

The Genetic Algorithm (GA) is considered one of the most widely researched methodologies. It was originally proposed by Holland [44] and uses the metaphor of survival of fittest. GAs have been implemented and applied in a wide spectrum of optimization problems [3]. An abstract description of a simple GA is given in Figure 3.3.

```
Initialize the population (p_0) of candidate solutions  s_i ∈ S
gen=0
While number of generations has not reached the maximum,
   gen=gen+1
   Create a set of solutions (m) via crossover  S_m^crossover  from p_gen-1
   Create a set of solutions (n) from S_m^crossover via mutation  S_n^mutation
       For each s_i ∈ S, S_m^crossover, S_n^mutation
         Evaluate the fitness
```

Figure 3.3: Implementation of a Generic GA

3.3.4. Artificial Immune Algorithm (IA)

Artificial Immune Algorithms (IA) have been implemented and applied in optimization problems, especially scheduling and bio-engineering applications [15]. IA uses the metaphor of the human immune system. The antibodies and antigens of the human body's defensive system are used for building the optimization model. Although the basic implementation of IA is similar to GA, the main differences between them lie in the manner the solutions are selected for diversification and intensification mechanisms using affinity computations [1]. An abstract description of an IA is given below in Figure 3.4.

Initialize the population (p) of candidate solutions $s_i \in S$
While number of generations has not reached the maximum,
Compute fitness function values $f(s_i)$
Compute Affinity values $affi(i) = \frac{1}{1+\frac{1}{k}\sum_{j=1}^{k} r_{ij}}$ where $r_{ij} = -p_{ij}\log(p_{ij})$
(k size of the sequence of the antibody, i being the solution in concern and j being the reference solution)
Use elite reproduction
Create the mating pool using Affinity threshold

Figure 3.4: Implementation of a Generic IA

For each algorithm, we tested three different neighborhoods on the permutations (local searches): inversion, transposition, and displacement (Figure 3.5). Our preliminary analyses of the simulation runs verified the findings of Dreo *et al.* [31]; the worst performance was observed for the inversion and displacement methods. Therefore, we implemented the transposition technique in all of the algorithms.

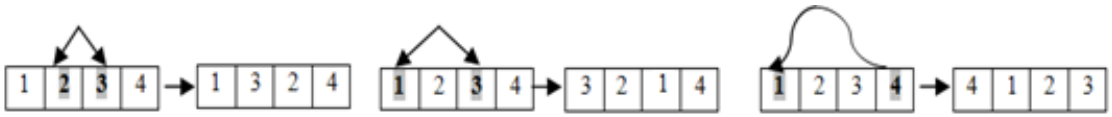


Figure 3.5: Neighborhoods on Permutations (Inversion, Transposition, and Displacement, respectively)

3.4. Input Data

We used 130 test cases from the QAPLIB problem repository (all test cases except *Tai150b, Tai256c*, and *Tho150*) ranging from problem sizes 12 to 128 (Please refer to Appendix A for a detailed description of the test cases.) These instances consisted of symmetric, asymmetric, and rectangular data from pseudorandom number generation and actual data collected from hospital layouts, backboard wiring problems, and many other real-world applications [17]. These problems have been either optimally solved by exact mathematical methods or approximated by metaheuristics. In order to utilize a unified performance measure, a loss-function value was computed. The loss-function is computed using the deviation of a particular solution from the best-known solution found in the literature. This can be computed using the following equation.

$$\text{Loss-function} = \frac{Solution - (Best - known)}{Best - known} \qquad (1)$$

When a particular problem was optimally solved, the optimal value was used. In all other cases, however, the best-known values in the literature were considered.

3.5. The Two Solution Representation Schemes

The solution representations of the algorithms utilize a random-keys representation following Bean [13] and a permutation representation following Tate and Smith [85]. These representations enable the exploration of solutions without violating the feasibility conditions. Figure 3.6 illustrates how crossover operations of solutions can generate infeasible solutions.

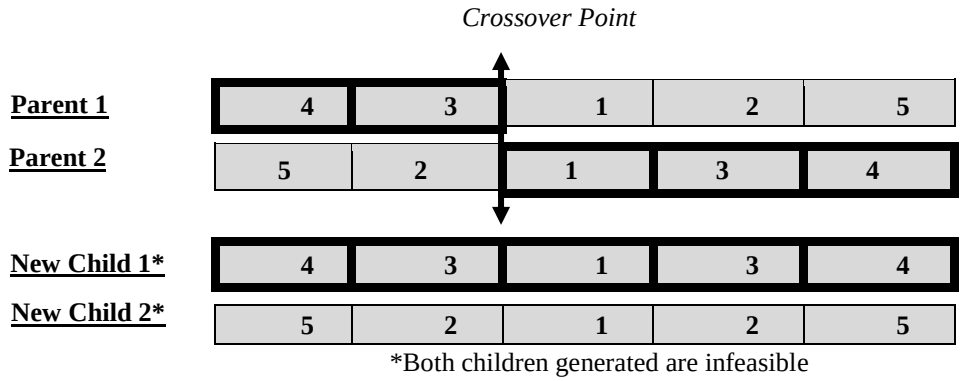
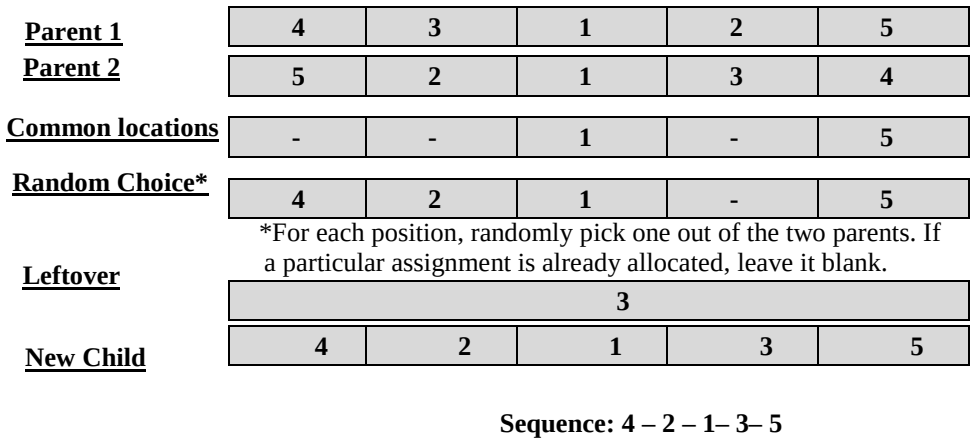


Figure 3.6: Infeasibility Issues with Permutation-Solution Representation

In order to overcome these issues, Tate and Smith [85] proposed a modified permutation representation scheme. The four metaheuristics were implemented using this solution encoding procedure. Figure 3.7 depicts how this method has addressed eliminating infeasibility through crossover operations.



Sequence: 4 – 2 – 1– 3– 5

Figure 3.7: New Solution Generation Using Permutation Representation

An alternative representation scheme to aid combinatorial optimization problems was first proposed in scheduling-related applications. This representation is well suited for permutation-based problems, as it helps ensure feasibility. Figure 3.8 illustrates a feasible solution generated by the random-keys solution in which the individuals are strings of real-valued numbers (random keys) in the interval of [0, 1]. These keys are sorted to generate the sequence or permutation of facilities to be assigned to locations (see Figure 3.8). In this study, all four of the metaheuristics were implemented using random-keys solution representations.

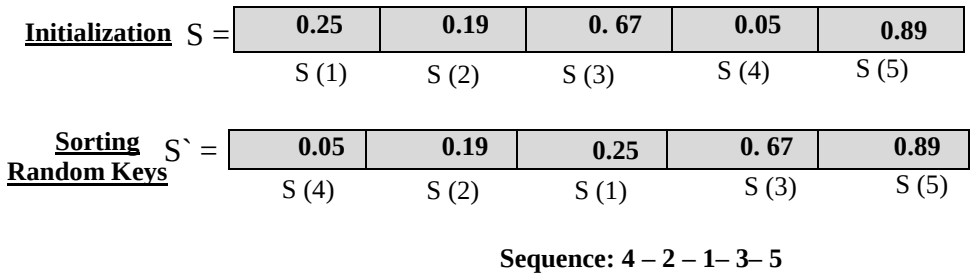


Figure 3.8: Illustration of Random-Keys Representation

3.6. High Throughput Computing (HTC)

Due to the difficulty of optimally solving complex QAPs, researchers often utilize high-performance computers or parallel processing [58]. Recently, James *et al.* [45] and James *et al.* [46] used sequential and parallel TS as well as cooperative TS for QAP. Anstreicher *et al.* [5] used a large-scale computational grid to optimally solve a few of the most well-known hard test cases using a branch-and-bound algorithm. High throughput computing (HTC) is an alternative to high-performance computing that operates with a geographically dispersed pool of computers, which are integrated via a communication network [57]. For our study, we have utilized a specific HTC environment—Condor—which is implemented at Clemson University

with 730 Windows and 771 Linux machines. The Windows machines have either dual-core or quad-core processors with the majority having 64-bit quad-core Intel Xeons with 3GHz processors and 8 gigabytes of RAM. The 730 Windows machines provide 2,370 cores, meaning that 2,370 jobs can be run simultaneously. The Linux machines can run up to 6,168 jobs simultaneously. These machines are a mixture of Intel and AMD processors with a clock rate of 2.3GHz per core and either 12 or 16 gigabytes of RAM.

For all four of the algorithms, each treatment on the experimental design was replicated 50 times for each input file (130 files) and for each representation scheme on Condor. The final loss-function values were computed by averaging the 50 replications. Each replication utilized pseudorandom numbers from independent non-overlapping streams, which were generated from a Mersenne Twister [62]. The first phase of the study required approximately 1,022,852 hours of computation time but actually took fewer than 60 days of clock time due to Condor's efficiency. The extensive computations carried out for this study were comparable to 116.6 years of computations on a single machine.

3.7. Performance Measures

In the literature, there are two performance measures that have been widely used to consider the efficiency of algorithms: the run time of the algorithm as a relative measure [11] and number-of-fitness evaluations as an absolute measure [12]. However, run times are highly correlated with the configurations of the machine. The Condor environment allows the use of dissimilar machines, so we prefer to utilize number-of-fitness evaluations as the performance measure. In this study, the effect of

the number-of-fitness evaluations on performance is first examined in the comparison study of path-based algorithms and is then evaluated in the population-based comparison. When the size of the population and the number of generations are varied, the total number of fitness evaluations is held as a constant.

In order to evaluate the solution quality of each method, a success rate was computed. Three measures were introduced: the number of times the best-known solution was found for each input file for each replication, the number of times a solution found within 1% of the best-known solution, and the number of times a solution found within 5% of the best-known solution. These measures will give insight into how each method performs when the average loss-function values show only marginal differences.

3.8. Parameter Tuning using Design of Experiments (DOE)

Barr *et al.* [11] and Coy *et al.* [20] indicate the importance of parameter tuning in evolutionary algorithms. Specifically, Coy *et al.* [20] utilized a four-step design of experiment-based procedure of a Vehicle Routing Problem (VRP). Further, Birattari [16] investigated the effects of parameter tuning from a machine-learning perspective, including solving QAP using a local search for one of the test cases. In recent literature, Adenso-Daz and Laguna [2] proposed a Taguchi fractional factorial experiment-based procedure called CALIBRA, which can tune up to five design factors. In addition, the following studies have used the design-of-experiment approach to investigate the impact of parameter tuning on performance using different problem domains. For example, Park and Kim [73] identified several design factors for SA using a nonlinear response-surface method with the Simplex method.

Using TS, Xu *et al.* [90] developed a five-design factor-tuning procedure for the Steiner Tree-Star problem. Finally, Deb and Agrawal [23] investigated four structural properties of problems related to testing GAs.

Based on the algorithmic descriptions above, we have selected several parameters for each metaheuristic of interest. Table 3.1 shows the range of parameter values considered for each design factor.

Table 3.1: Parameters for the Experimental Design of the Four Metaheuristics

Metaheuristic	**Parameter**	**Range of the Parameters**
SA	Initial Temperature	10^{-3} , 10^{-2}, 10^{-1}, 10, 10^{3}, 10^{5}
	Cooling Schedule	0.3, 0.5, 0.80, 0.95, 0.99
	Number of Iterations	10, 10^{3}, 10^{5}, 10^{6} ,10^{9}
TS	Tabu List	5, 10, 15, 20, 25, 30
	Neighborhood Size	10, 15, 20, 25, 30
	Number of Iterations	10, 10^{3}, 10^{5}, 10^{6}, 10^{9}
GA	Population Size	10, 10^{2}, 10^{3}, 10^{4}
	Crossover Probability	0.5, 0.8, 0.9
	Mutation Probability	0.15, 0.5, 0.8 ,0.9
	Number of Generations	10, 10^{2}, 10^{3}, 10^{4}, 10^{5}
IA	Population Size	10^{2}, 10^{3}, 10^{4}
	Crossover Probability	0.5, 0.8, 0.9
	Mutation Probability	0.15, 0.5, 0.8, 0.9
	Affinity Threshold	0.3, 0.5, 0.8, 0.9, 0.98
	Affinity Adjustment	0.01, 0.3, 0.5, 0.8
	Number of Generations	10, 10^{2}, 10^{3}, 10^{4}, 10^{5}

Each metaheuristic with the respective representation scheme was replicated 50 times using 130 input files. The average loss-function values, variances, and success rates were compared using nonparametric/parametric statistical analysis techniques.

3.9. Experimental Results

The experimental results are presented separately for each representation scheme. Depending on the representation scheme, the most appropriate parameters for each design factor varied. In the full factorial design, the average loss-function values over the 50 replications were compared for each treatment. The treatments that showed very similar performance were discarded, while for graphical visualizations, the treatments with significant differences were presented.

3.9.1 Simulated Annealing

The metaheuristic design factors considered for SA included the initial temperature, the cooling schedule, and the number of iterations. See Figure 3.9 and Table 3.2 for selected treatments of SA for each representation.

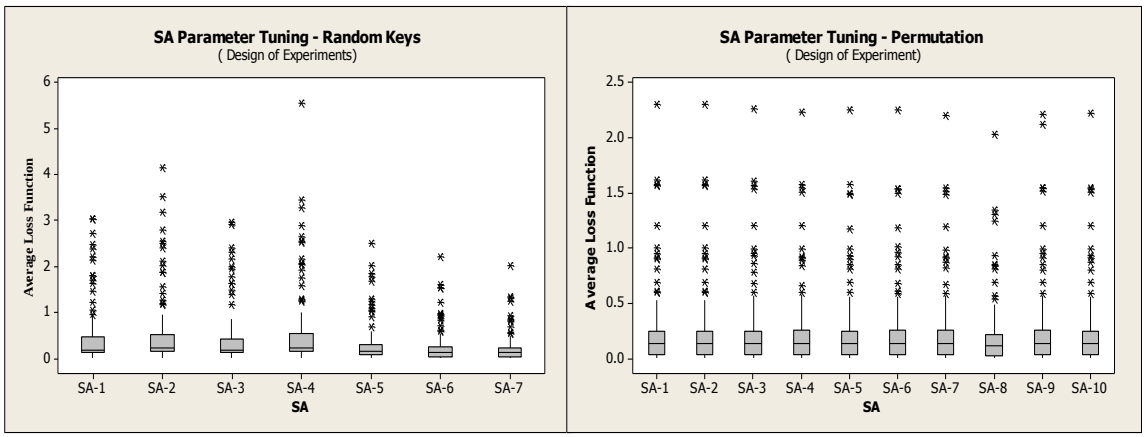


Figure 3.9: Box-Plots for SA with Different Treatments Using Two Representations

Table 3.2: SA with Different Treatments Using Two Representations

Representation Scheme	Treatment Number	Parameters (Temperature, Cooling factor, Number of Iterations)
Random Keys	SA-1	10^3,0.99,10^3
	SA-2	10^5,0.95,10^5
	SA-3	10^5,0.99,10^3
	SA-4	10^5,0.99,10^5

	SA-5	10^5,0.90,10^6
	SA-6	10^3,0.8,10^3
	SA-7	10^5,0.99,10^3
Permutation	SA-1	10^5,0.90,10^6
	SA-2	10^4,0.90,10^6
	SA-3	10^3,0.90,10^6
	SA-4	10^5,0.80,10^6
	SA-5	10^3,0.80,10^6
	SA-6	10^2,0.80,10^6
	SA-7	10^5,0.50,10^6
	SA-8	10^4,0.50,10^6
	SA-9	10^3,0.50,10^6
	SA-10	10^2,0.50,10^6

It was found that as the number of iterations increases, the average loss-function values were significantly improved. However, in order to carry out a fair comparison, the number of iterations that a particular metaheuristic was executed remained constant. For both representations, SA performed well with high initial temperatures and low cooling schedules irrespective of the number of iterations, whereas, low initial temperatures and low cooling schedules had an adverse effect on performance. When the initial temperature was reduced up to a threshold, holding the cooling schedule and the number of iterations constant, the loss-function values significantly improved. It was found that for SA, the best treatment for the random keys and the permutations were the same. In other words, even though there is a statistically significant difference in the parameter-tuning process for SA for the different representations, once tuned, the effects of the representation scheme had no effect on loss function. Figure 3.10 depicts box plots with the average loss-function values and problem size. As shown, treatment number seven performs better with respect to these two measures. It can be seen that for all the treatments, most of the outliers are clustered around problem sizes >100.

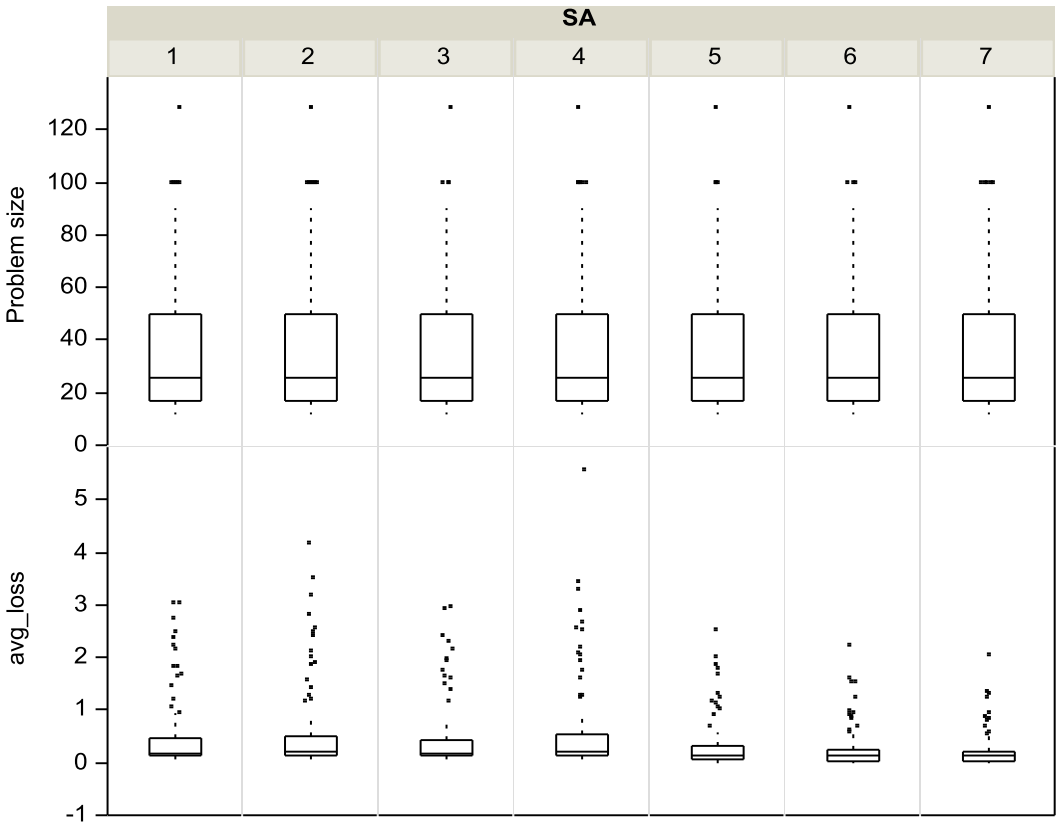


Figure 3.10: Box-Plots with Average Loss and Problem Size for SA with Random Keys

Out of the selected treatments that showed significant differences among their average loss-function values, the best treatment was selected using the Mood median test and nonparametric multiple comparison among the medians. Figure 3.11 depicts the confidence intervals of these treatments. Using the Mood median test results, the tuned treatments were selected. As seen in Figure 3.10, for the random-keys representation (left), treatment number seven had the lowest median value. The same procedure was applied for the permutation-solution representation, and treatment seven was again selected.

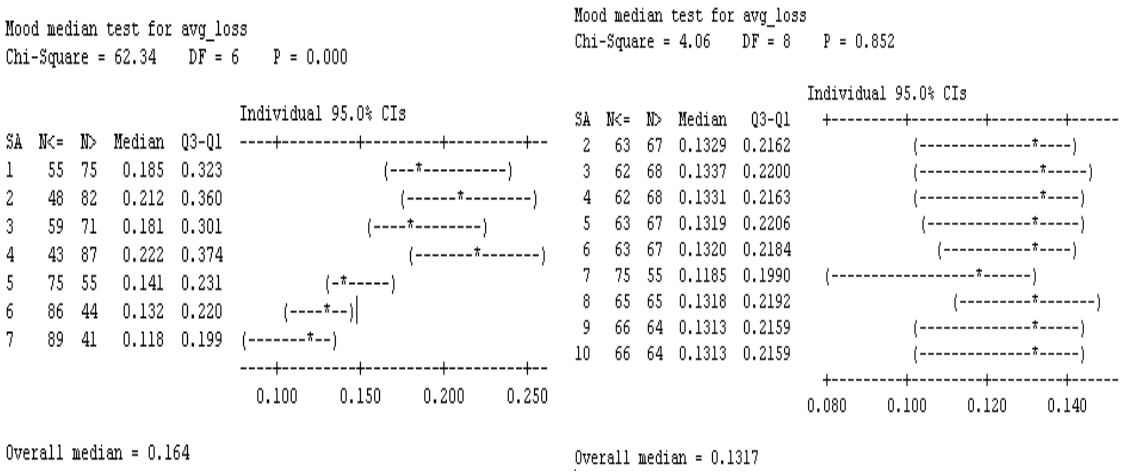


Figure 3.11: Mood Median Test for SA with Random-Keys (left) permutation (right)

3.9.2 Tabu Search

The metaheuristic design factors considered for TS include tabu list, neighborhood size and number of iterations. Similar to SA, when number of iterations increased the loss-function values improved. The number of iterations used for SA was also used for TS, but other design factors varied.

As one would expect, when the size of the neighborhood increased, the search space expanded, thus decreasing the loss function. This phenomenon was true for the random keys; however, when the neighborhood size was increased from 10 to 30 (10, 15, 20, 25, and 30), significant improvement was not shown at 30 for the permutation representation. The tabu list, size varied from 5 to 20, and the best parameter was 10 for the random keys and 15 for the permutation. When the neighborhood size and number of iterations were held constant, increasing the tabu list adversely affected the loss function. Furthermore, smaller tabu lists created favorable outcomes for the loss function. It was noted that an interaction effect exists between the size of the

neighborhood and the size of the tabu list and careful consideration should be given to these parameters. Since, major changes in one metaheuristic factor cause the other factor to drastically affect performance adversely. Figure 3.12 and Table 3.3 depict the box-plots of the TS for each representation.

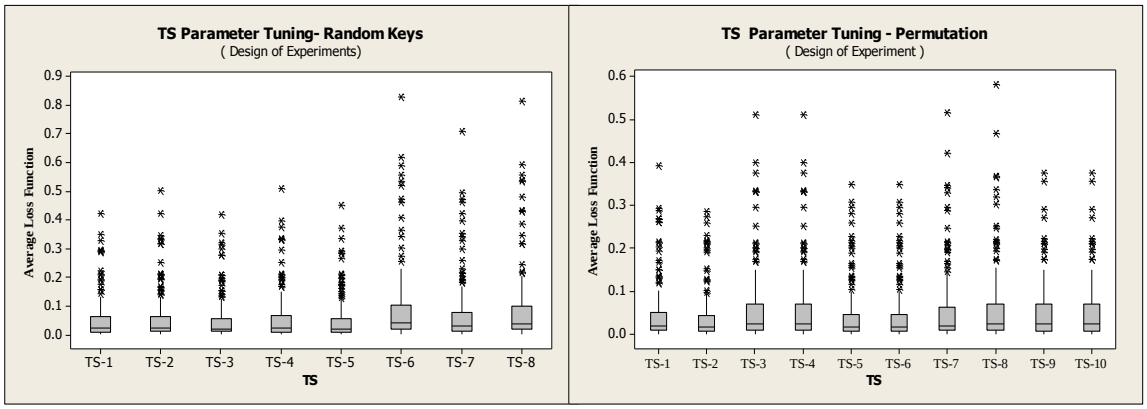


Figure 3.12: Box-Plots for TS with Different Treatments Using Two Representations

Table 3.3: TS with Different Treatments Using Two Representations

Representation Scheme	Treatment Number	Parameters (Size of Neighbors, Tabu List, Number of Iterations)
Random Keys	TS-1	$20,10,10^5$
	TS -2	$25,15,10^5$
	TS -3	$25,05,10^3$
	TS -4	$20,20,10^5$
	TS -5	$30,10,10^6$
	TS -6	$20,20,10^5$
	TS -7	$30,10,10^3$
	TS -8	$30,10,10^2$
Permutation	TS -1	$30,10,10^2$
	TS -2	$20,15,10^6$
	TS -3	$20,05,10^6$
	TS -4	$25,10,10^6$
	TS -5	$25,15,10^5$
	TS -6	$25,05,10^6$
	TS -7	$30,10,10^6$
	TS -8	$30,15,10^3$
	TS -9	$30,05,10^6$
	TS -10	$30,20,10^6$

The Mood median test was used again to select the treatment with the lowest median value. From the statistical analysis, it was concluded that the permutation solution representation was more suitable for TS than the random-keys representation. Figure 3.13 depicts the performance of each TS treatment with various problem sizes.

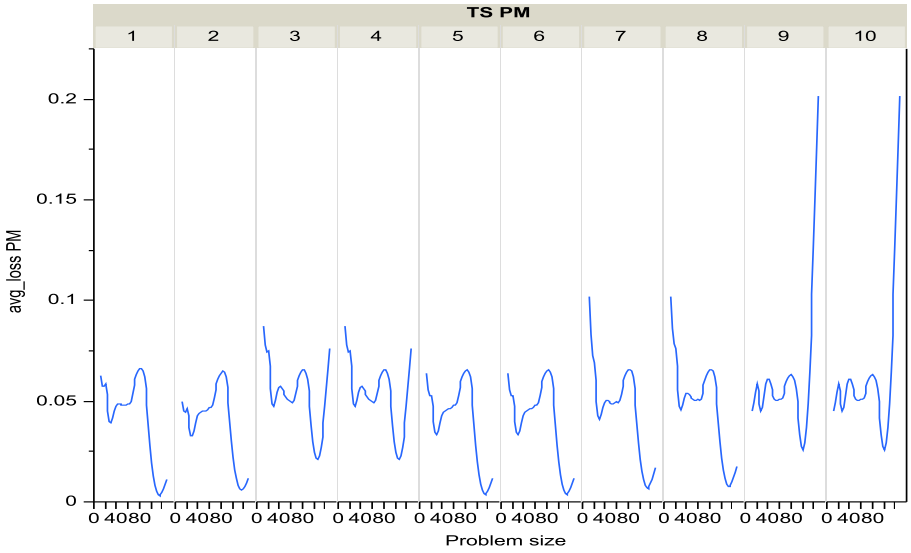


Figure 3.13: Performance of TS with the Permutations for Different Problem Sizes

Figure 3.14 shows the confidence intervals and the median values of the TS for each representation. From these comparative statistical analyses, TS permutation treatment number five was selected as the lowest median loss-function value.

Mood median test for avg_loss
Chi-Square = 79.57 DF = 7 P = 0.000

TS	N<=	N>	Median	Q3-Q1	Individual 95.0% CIs
1	77	53	0.0217	0.0531	(---*-----)
2	75	55	0.0235	0.0500	(---*-------)
3	80	50	0.0198	0.0467	(--*------)
4	72	58	0.0244	0.0589	(----*-----)
5	84	46	0.0194	0.0478	(--*-----)
6	36	94	0.0407	0.0850	(---*---------)
7	60	70	0.0302	0.0667	(----*------)
8	36	94	0.0389	0.0777	(----*--------)

0.020 0.030 0.040 0.050

Overall median = 0.0284

Mood median test for avg_loss
Chi-Square = 21.54 DF = 9 P = 0.010

TS	N<=	N>	Median	Q3-Q1	Individual 95.0% CIs
1	69	61	0.0178	0.0407	(---*---------)
2	75	55	0.0157	0.0360	(----*------)
3	53	77	0.0244	0.0608	(--------*--------)
4	53	77	0.0244	0.0608	(--------*--------)
5	76	54	0.0157	0.0379	(---*-----)
6	76	54	0.0157	0.0379	(---*-----)
7	65	65	0.0193	0.0539	(---*----------)
8	59	71	0.0223	0.0604	(------*---------)
9	62	68	0.0226	0.0631	(----------*----------)
10	62	68	0.0226	0.0631	(----------*----------)

0.0180 0.0240 0.0300

Overall median = 0.0191

Figure 3.14: Mood Median Test for TS with Random Keys (left) and Permutation (right)

3.9.3 Genetic Algorithms

Metaheuristic design factors for GAs include population size, crossover probability, mutation probability, and number of generations. In this study, as the number of generations increased, the loss-function values significantly improved (Figure 3.15 and Table 3.4). In order to compare the loss-function values of SA and TS, the number of fitness evaluations (number of generations X population size) were held constant.

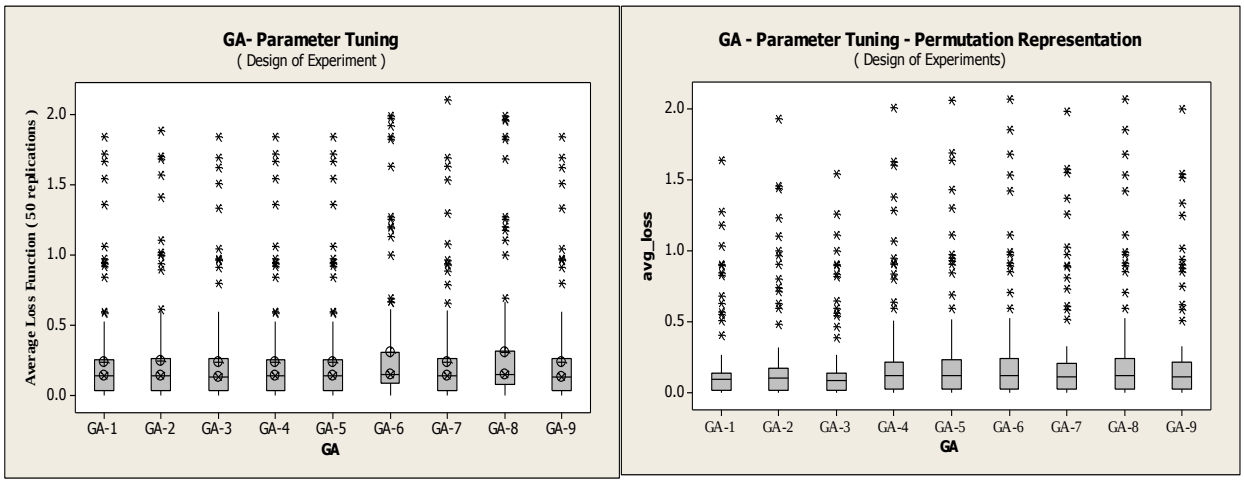


Figure 3.15: Box-Plots for GA with Different Treatments Using Two Representations

Table 3.4: GA with Different Treatments Using Two Representations

Representation Scheme	Treatment Number	Parameters (Population Size, Generations, Crossover, Mutation)
Random Keys	GA-1	10^3, 10^3,0.8,0.15
	GA -2	10^3, 10^3,0.5,0.15
	GA -3	10^3, 10^3,0.9,0.15
	GA -4	10^3, 10^3,0.8,0.5
	GA -5	10^3, 10^3,0.8,0.8
	GA -6	10^2, 10^4,0.8,0.8
	GA -7	10^4, 10^2,0.8,0.8

	GA -8	10^2, 10^4,0.9,0.9
	GA -9	10^4, 10^2,0.9,0.9
Permutation	GA -1	10^3, 10^3,0.8,0.15
	GA -2	10^3, 10^3,0.5,0.15
	GA -3	10^3, 10^3,0.9,0.15
	GA -4	10^3, 10^3,0.8,0.5
	GA -5	10^3, 10^3,0.8,0.8
	GA -6	10^2, 10^4,0.8,0.8
	GA -7	10^4, 10^2,0.8,0.8
	GA -8	10^2, 10^4,0.9,0.9
	GA -9	10^4, 10^2,0.9,0.9

Unlike the path-based metaheuristics, the population-based methods were less affected by the parameter-tuning process for a given representation scheme. However, for GA, the permutation representation was better suited than the random–keys representation. For the random-keys representation, GA performed well with a higher number of generations, a higher crossover, and higher mutation probabilities. For the permutation representation, the population size was equally important as the number of generations and the higher crossover probability, but a very low mutation probability managed to find the lowest loss-function values unlike in the other treatments. For both representations, higher crossover probabilities resulted in lower loss-function values. In addition, the mutation probability needed to be less than the crossover probability for each representation, meaning that an iteration effect exists between the two. Figure 3.16 depicts the Mood median test results for the selected treatments of the two GAs and the respective median values and confidence intervals.

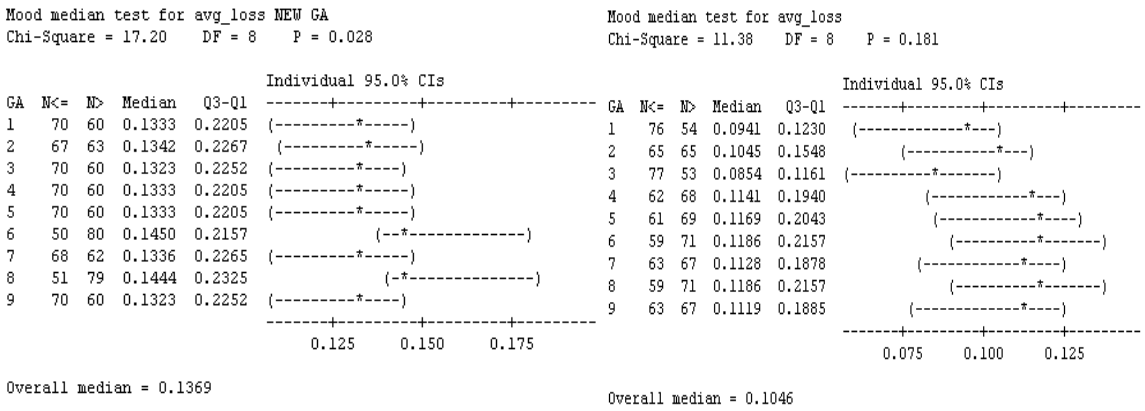


Figure 3.16: Mood Median Test for GA with Random Keys (left) and Permutation (right)

3.9.4 Immune Algorithm

Similar to GAs, for artificial Immune Algorithms, the metaheuristic design factors considered include population size, crossover and mutation probabilities, and the number of generations. The design factors that are specific to IA include affinity threshold and affinity adjustment. These design factors allow IAs to diversify and simultaneously control the diversification to prohibit the over-dominance of certain good solutions in a particular IA population. From Figure 3.17 and Table 3.4, it is evident that the random keys representation creates less variability within the parameter-tuning process as opposed to the permutation-representation scheme. Hence, the statistical analysis showed that the permutation representation outperformed the random keys representation (p<0.05). Selected treatments are depicted in Figure 3.17 and Table 3.5) for each representation.

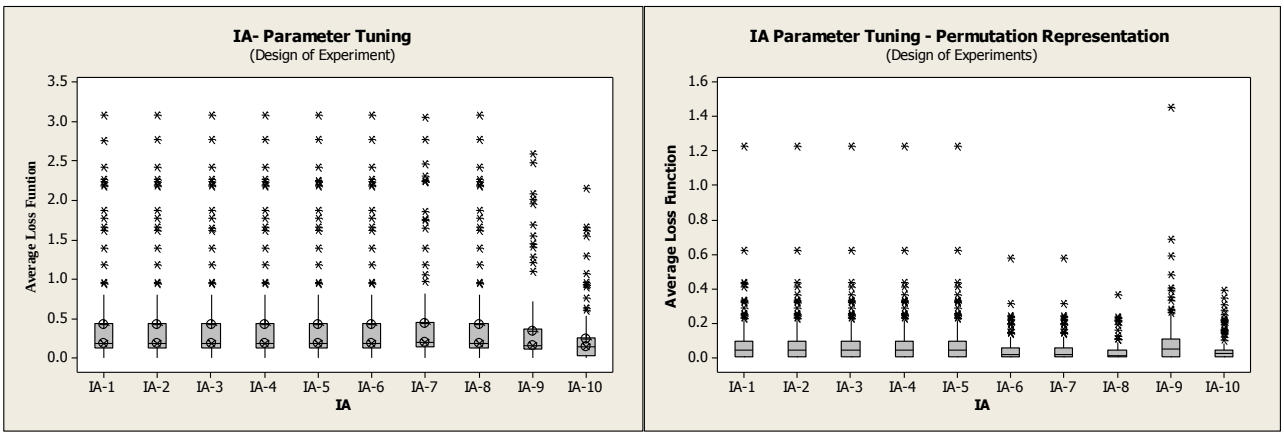


Figure 3.17: Box-Plot for IA with Different Treatments Using Two Representations

Table 3.5: IA with Different Treatments Using Two Representations

Representation Scheme	Treatment Number	Parameters (Population Size, Generations, Crossover, Mutation, Affinity Threshold, Affinity Adjustment)
Random Keys	IA-1	10^3, 10^3,0.5,0.15,0.98,0.5
	IA -2	10^3, 10^3,0.5,0.15,0.98,0.3
	IA -3	10^3, 10^3,0.5,0.15,0.98,0.05
	IA -4	10^3, 10^3,0.5,0.15,0.90,0.5
	IA -5	10^3, 10^3,0.5,0.15,0.98,0.8
	IA -6	10^3, 10^3,0.5,0.5,0.98,0.5
	IA -7	10^3, 10^3, 0.9,0.5,0.98,0.8
	IA -8	10^3, 10^3, 0.5,0.9,0.98,0.8
	IA -9	10^4, 10^2, 0.9,0.9,0.98,0.8
	IA-10	10^2, 10^4, 0.9,0.9,0.98,0.8
Permutation	IA -1	10^3, 10^3, 0.5,0.15,0.98,0.5
	IA -2	10^3, 10^3, 0.5,0.15,0.98,0.3
	IA -3	10^3, 10^3, 0.5,0.15,0.98,0.05
	IA -4	10^3, 10^3, 0.5,0.15,0.90,0.5
	IA -5	10^3, 10^3, 0.5,0.15,0.98,0.8
	IA -6	10^3, 10^3, 0.5,0. 5,0.98,0.5
	IA -7	10^3, 10^3, 0.9,0. 5,0.98,0.8
	IA -8	10^3, 10^3, 0.5,0.9,0.98,0.8
	IA -9	10^4, 10^2, 0.9,0.9,0.98,0.8
	IA-10	10^2, 10^4, 0.9,0.9,0.98,0.8

Some interesting observations were noted based on these analyses. For both representations, the affinity thresholds and adjustments required higher values to generate competitive loss-function values. For the random-keys representation, a higher crossover probability and a lower mutation probability were more appropriate, while for the permutation representation, the opposite held true. Figure 3.18 shows the Mood median test results for IA for both representations.

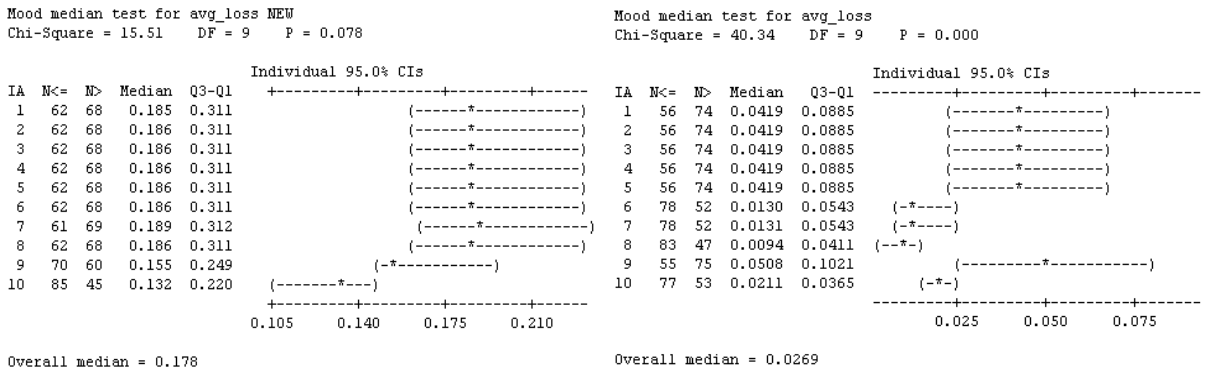


Figure 3.18: Mood Median Test for IA with Random Keys (left) and Permutation (right)

3.10 Comparison Study using Parametric and Non-Parametric Statistics

Since the loss function values measure the deviation from the best-known value, the distributions of the values are skewed a great deal; hence, nonparametric statistical analysis techniques are more suitable than parametric techniques. Thorough analysis of the residual plots further confirms this phenomenon. However to improve the power of the statistical analysis, parametric (one-way Analysis of Variance: ANOVA) as well as non-parametric statistical analysis (Friedman's test and Mood median test) techniques were utilized. When the two methods' results

contradict, three underlying assumptions were checked for normality (independence, distribution of the residual plots, and the homogeneity or equality of the variances). Table 3.6 depicts the summary of the statistical analyses for each metaheuristic.

Table 3.6: Summary of the Parametric and Non-Parametric Statistical Analyses

Metaheuristic	Representation	Statistical Analysis (Significance of Parameter Tuning)		Conclusion
		One-Way ANOVA	Friedman's Test	
SA	Random Keys	**Significant (p<0.05)**	**Significant (p<0.05)**	**Significant**
	Permutation	**Significant (p<0.05)**	Not Significant (p>0.05)	Not Significant
TS	Random Keys	**Significant (p<0.05)**	**Significant (p<0.05)**	**Significant**
	Permutation	**Significant (p<0.05)**	**Significant (p<0.05)**	**Significant**
GA	Random Keys	Not Significant (p>0.05)	Not Significant (p>0.05)	Not Significant
	Permutation	Not Significant (p>0.05)	Not Significant (p>0.05)	Not Significant
IA	Random Keys	Not Significant (p>0.05)	Not Significant (p>0.05)	Not Significant
	Permutation	**Significant (p<0.05)**	**Significant (p<0.05)**	**Significant**

After analyzing the significance of parameter tuning on each metaheuristic, a comparative analysis has been carried out among the best tuned representation for each method. Table 3.7 depicts the results of the summary of the comparative analysis for each metaheuristic.

Table 3.7: Summary of the Comparative Analysis.

Metaheuristic	Best Representation	Comparative Statistical Analysis (selecting the best)		Conclusion
		One-way ANOVA	Friedman's test	
SA	Random Keys/ Permutation (indifferent)	Not Significant (p>0.05)	Not Significant (p>0.05)	Not Significant

TS	Permutation	Not Significant (p>0.05)	**Significant (p<0.05)**	**Significant**
GA	Permutation	**Significant (p<0.05)**	**Significant (p<0.05)**	**Significant**
IA	Permutation	**Significant (p<0.05)**	**Significant (p<0.05)**	**Significant**

For the four tuned metaheuristics that the Friedman's test found, IA with permutation outperformed the rest (p<0.05). One interesting observation was that even though the IA-based random-keys metaheuristic was the worst among the eight initial treatments, it significantly improved with permutation representation scheme. Hence, permutation-based IA outperformed the rest.

Table 3.8 summarizes the best and worst cases of each method's parameter-tuning procedures. The comparative measures include average loss-function value, standard deviation, number of times the best-known solution was found, and the number of times the solutions found within 1% and 5% of the best-known solution.

Table 3.8: Best and Worst Case Scenarios of Parameter Tuning (130 files and 50 replications)

Method	Random Keys				Permutation			
	Average (Standard Deviation)	Percentage of Number of Times within 1% of Best-Known Solution			Average (Standard Deviation)	Percentage of Number of Times within 5% of Best-Known Solution		
		0%	1%	5%		0%	1%	5%
SA-Best	0.1911 (0.2975)	10.74%	13.63%	**33.68%**	0.1911 (0.2970)	10.70%	13.60%	**33.68%**
SA-Worst	0.5309 (0.8221)	1.65%	1.83%	12.91%	0.22705 (0.3468)	8.98%	10.17%	29.78%
TS-Best	0.0546 (0.0840)	17.65%	35.92%	74.94%	0.0004 (0.0006)	23.29%	43.22%	**80.05%**
TS-Worst	0.0990 (0.1472)	10.25%	22.34%	57.92%	0.0006 (0.0009)	16.00%	32.28%	72.22%
GA-Best	0.0023 (0.0034)	8.11%	8.57%	28.54%	0.0015 (0.0026)	15.60%	21.88%	**39.12%**

GA-Worst	0.0030 (0.0043)	5.06%	5.55%	23.74%	0.0022 (0.0037)	13.86%	14.22%	33.37%
IA-Best	0.0023 (0.0035)	8.22%	8.85%	28.78%	0.0001 (0.0005)	36.78%	53.03%	**82.43%**
IA-Worst	0.0042 (0.0060)	2.12%	2.63%	16.78%	0.0009 (0.0016)	18.69%	30.55%	50.31%

From Table 3.4, it was evident that although the best SA (treatment that gave the lowest loss-function value) was not affected by the representation scheme, when the worst two treatments were compared, the permutation representation performed better than the random-keys representation, and the success rate also improved.

When TS-best and TS-worst were compared, the permutation representation outperformed the random-keys representation for each measure. It was noted that TS-worst with permutation was as competitive as TS-best with random keys; hence, the importance of selecting a suitable representation scheme is highlighted. This phenomenon is also true for both GA and IA. From Table 3.4, it was noted that selecting the most suitable solution representation and parameter-tuning procedure can significantly improve solution quality, up to 80% for TS and IA.

3.11 Categorization of QAPLIB Input files

For the parameter-tuning procedure, we used 130 input files from the QAPLIB problem repository dedicated to QAP related research. Table 3.9 presents the summary of those problem instances, including the domain, problem size, authors, and data-generation procedures.

Table 3.9: Summary of the QAPLIB Input files

#	Number of files	Problem size	Name	Authors	Context	Remarks
1	9	26	Bur26*	R.E. Burkard and J. Offermann	Real world	Asymmetric matrices
2	14	12 to 25	Chr*	N. Christofides and E. Benavent	Random	One matrix is the adjacency matrix of a weighted tree; the other is that of a complete graph
3	1	19	Els19	A.N. Elshafei	Real world	Hospital and the flow of patients between those locations
4	19	16 to 128	Esc*	B. Eschermann and H.J. Wunderlich	Real world	Application in computer science from the testing of self-testable sequential circuits
5	5	12 to 20	Had*	S.W. Hadley, F. Rendl and H. Wolkowicz	Random	Manhattan distances of a connected cellular complex in the plane; flow matrix are drawn uniformly from the interval [1,n]
6	3	30 to 32	Kra*	J. Krarup and P.M. Pruzan	Real world	The instances contain real-world data and were used to plan the Klinikum Regensburg in Germany
7	16	20	Lipa*	Y. Li and P.M. Pardalos	Random	The generators provide asymmetric instances with known optimal solutions
8	14	12	Nug*	C.E. Nugent, T.E. Vollmann and J. Ruml	Random	The distance matrix contains Manhattan distances of rectangular grids
9	3	12 to 20	Rou*	C. Roucairol	Random	The entries of the matrices are chosen from the interval [1,100]
10	3	12 to 20	Scr*	M. Scriabin and R.C. Vergin	Random	The distances of these problems are rectangular
11	13	42 to 100	Sko*	J. Skorin-Kapov	Random	The distances of these problems are rectangular, and the entries in the flow matrices are pseudorandom numbers
12	3	36	Ste*	L. Steinberg	Real world	The three instances model the backboard wiring problem; the distances in the first one are Manhattan and Euclidean
13	11	12 to 100	Taib*	E.D. Taillard	Real world	Taixxb are asymmetric
13	13	12 to 100	Taia*	E.D. Taillard	Random	Tai*xx*a are uniformly randomly generated
14	3	30 to 50	Tho30	U.W. Thonemann and A. Bölte	Random	The distances of these instances are rectangular
Total Number of Input Files = 130						

Out of the 130 files, 46 problem instances are real-world problems. These problems have been accumulated from backboard wiring problems, computer science applications, circuit board designs, and many facility location problems, including hospital layouts. The parameter-tuning procedure was extended to analyze how each

tuned metaheuristic would behave depending on the domain of the input files (real-world problems or pseudo-randomly generated problems). Figure 3.19 depicts how each of the eight tuned metaheuristics performed for the real-world problems.

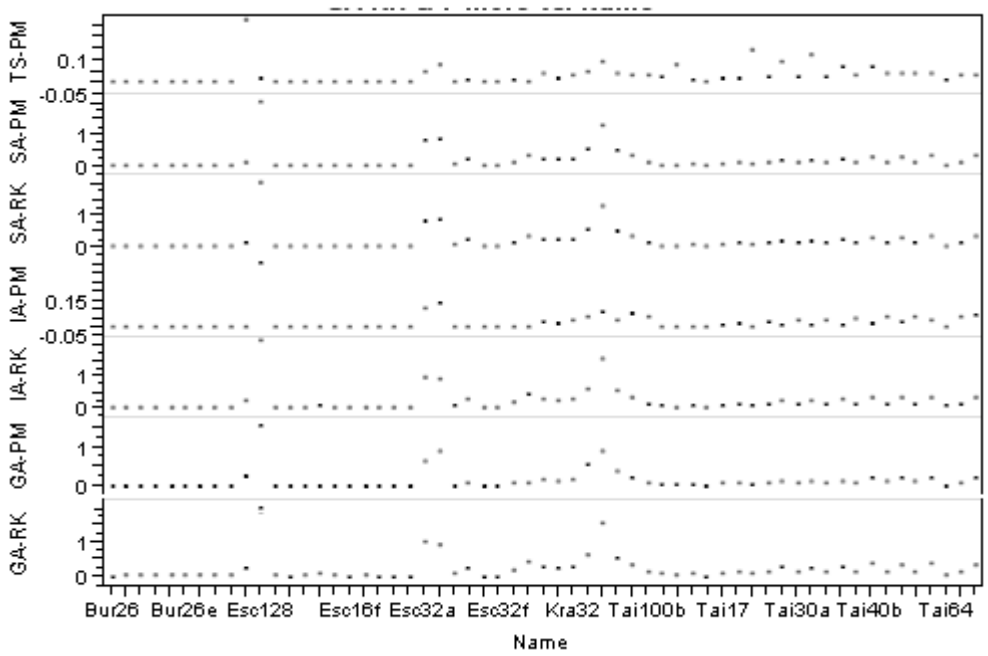


Figure 3.19: Eight Tuned Metaheuristics with 46 Real-World Problems

Irrespective of the method or the representation, it is clearly evident from Figure 3.19 that some problem instances were trivial. Therefore, these problem instances (Bur* and some of the Esc*) were discarded and were not included in further analyses. Figure 3.20 depicts how each eight of the tuned metaheuristics performed for the pseudo-randomly generated problems. This visualization includes 84 input files.

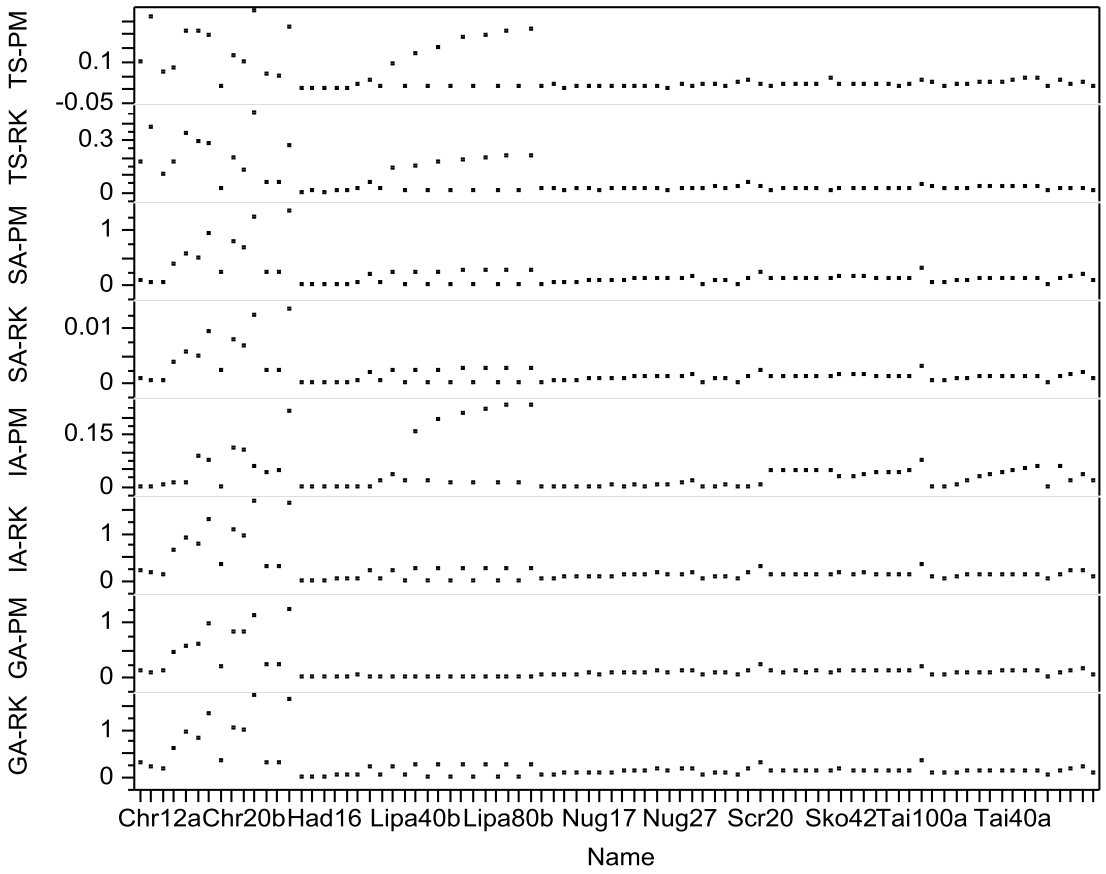


Figure 3.20: Eight Tuned Metaheuristics with 84 Pseudo-Randomly Generated Problems

From Figure 3.20 it is evident that a few of the problems instances were trivial (Had*, Rou* and some Nug* and Tai* problems). Similar to the real-world problems, in this case some clusters of the hard problems were visible. These problems were considered for further analyses in later chapters. Table 3.10 depicts the categorization of the input files based on the parameter-tuning procedures, including hard, moderate, and trivial cases for further analyses.

Table 3.10: Summary of the QAPLIB Input Files After Classification
(Optimally Solved Problems are Boldfaced)

Category	Average loss-function value	Number of files	File Names
Trivial Problems for Metaheuristics	< 5%	53	• **Bur26a** Bur26b Bur26c Bur26d Bur26e Bur26f Bur26g Bur26h Bur26 • **Esc16a Esc16b Esc16c Esc16d Esc16e Esc16f Esc16g Esc16h Esc16i Esc16j** Esc32c Esc32e Esc32f • **Had12 Had14 Had16 Had18 Had20** • **Lipa20a Lipa30a Lipa40a Lipa50a Lipa60a Lipa70a Lipa80a Lipa90a** • **Nug12 Nug14 Nug15 Nug16a Nug16b Nug17 Nug18** • **Rou12 Rou15 Rou20** • **Scr12** • **Tai12a Tai12b Tai15a Tai15b Tai17** Tai64 Tho50
Moderately Difficult Problems for Metaheuristics	5%< >20%	63	• **Chr12a Chr12b Chr12c Chr18b Chr22a Chr22b** • **Els16** Esc32d Esc32g Esc64a • **Kra30a Kra30b Kra32** • **Lipa20b Lipa30b Lipa40b Lipa50b Lipa60b Lipa70b Lipa80b Lipa90b** • **Nug20 Nug21 Nug22 Nug24 Nug25 Nug27 Nug30** • **Scr15 Scr20** • Sko42 Sko49 Sko56 Sko64 Sko72 Sko81 Sko90 Sko100a Sko100b Sko100c Sko100d Sko100e Sko100f • Tai20a Tai20b Tai25a Tai25b Tai30a Tai30b Tai35a Tai35b Tai40a Tai40b Tai50a Tai50b Tai60a Tai60b Tai80a Tai80b Tai100b Tai100a • **Tho30** Tho40
Hard Problems for Metaheuristics	>20%	14	• **Chr15a Chr15b Chr15c Chr18a Chr20a Chr20b Chr20c Chr22c** • Esc32a Esc32b Esc128 • **Ste36a Ste36b Ste36c**
Total Number of Files		130 Files	

The thorough parameter-tuning process allowed us to differentiate trivial problems from more complex instances. We found that more than 53 problem instances can be easily solved by careful parameter setting.

In [83], the authors considered four categories of problem instances depending on the way the problems are generated. They classify 37 problems extracted from

QAPLIB as class I (random problems with entries uniformly distributed), class II (random flows on Manhattan grids), class III (real-life instances stemming from practical applications), and class IV (instances randomly created to resemble real-life problems). The authors compared the runtimes and solution quality of their newly proposed iterated local search-based algorithm (ES) with two state-of-the-art metaheuristics, namely Robust Tabu Search (Ro-TS) and Ant System (MMAS) [83]. We compared the solutions obtained from this extensive parameter-tuning procedure with the above three metaheuristic procedures to determine the solution quality and how far one method can be improved by merely adjusting the tuning process rather than augmenting any greedy, adaptive or other functional procedures.

Tables 3.11-3.14 depict the eight tuned metaheuristics (average loss-function values over 50 replications) and class I through IV problem instances according to [83].

Table 3.11 Eight Tuned Metaheuristics and Class I Problem Instances

Name	GA-PM	GA-RK	IA-RK	IA-PM	SA-RK	SA-PM	TS-RK	TS-PM	AVG	Category According to Table 3.6	Category According to Stutzle (2006)
Tai20a	9.53%	11.36%	11.49%	**1.43%**	9.83%	9.83%	2.56%	1.97%	7.25%	Moderate	**Class I**
Tai25a	9.98%	12.06%	12.22%	2.76%	10.72%	10.72%	2.65%	**2.31%**	7.93%	Moderate	
Tai30a	9.79%	12.09%	11.97%	3.15%	10.89%	10.89%	2.80%	**2.29%**	7.98%	Moderate	
Tai35a	10.67%	12.88%	13.03%	3.90%	12.12%	12.12%	**2.65%**	2.76%	8.77%	Moderate	
Tai40a	11.12%	13.28%	13.24%	4.58%	12.53%	12.53%	**2.91%**	3.12%	9.16%	Moderate	
Tai50a	11.46%	13.65%	13.69%	5.31%	13.00%	13.00%	**3.19%**	3.59%	9.61%	Moderate	
Tai60a	11.27%	13.43%	13.40%	5.69%	12.85%	12.85%	**3.15%**	3.76%	9.55%	Moderate	
Tai80a	10.01%	11.89%	11.85%	5.44%	11.57%	11.57%	**2.57%**	3.32%	8.53%	Moderate	

Table 3.12 Eight Tuned Metaheuristics and Class II Problem Instances

Name	GA-PM	GA-RK	IA-RK	IA-PM	SA-RK	SA-PM	TS-RK	TS-PM	AVG	Category According to Table 3.6	Category According to Stutzle (2006)
Nug30	11.92%	16.23%	16.41%	1.59%	14.45%	14.45%	1.65%	**1.25%**	9.74%	Moderate	**Class II**
Sko42	12.51%	15.88%	16.22%	2.65%	14.95%	14.95%	1.66%	**1.57%**	10.05%	Moderate	
Sko49	11.25%	15.17%	15.21%	2.80%	14.33%	14.33%	1.70%	**1.68%**	9.56%	Moderate	
Sko56	12.00%	15.42%	15.63%	3.62%	14.77%	14.77%	1.72%	**1.56%**	9.94%	Moderate	
Sko64	11.23%	14.58%	14.62%	3.81%	13.79%	13.79%	1.64%	**1.61%**	9.38%	Moderate	
Sko72	11.01%	14.32%	14.34%	4.01%	13.82%	13.82%	1.63%	**1.57%**	9.31%	Moderate	
Sko81	10.48%	13.71%	13.78%	4.07%	13.38%	13.38%	1.48%	**1.27%**	8.95%	Moderate	
Sko90	10.61%	13.69%	13.71%	4.29%	13.36%	13.36%	1.56%	**1.37%**	8.99%	Moderate	
Sko100a	10.47%	13.18%	13.13%	4.38%	12.77%	12.77%	1.36%	**1.26%**	8.66%	Moderate	

Table 3.13 Eight Tuned Metaheuristics and Class III Problem Instances

Name	GA-PM	GA-RK	IA-RK	IA-PM	SA-RK	SA-PM	TS-RK	TS-PM	AVG	Category According to Table 3.6	Category According to Stutzle (2006)
Bur26a	0.90%	1.92%	1.84%	**0.03%**	1.46%	1.46%	0.20%	0.16%	0.99%	Trivial	**Class III**
Bur26b	0.89%	1.89%	1.92%	**0.01%**	1.47%	1.47%	0.32%	0.20%	1.02%	Trivial	
Bur26c	1.01%	2.08%	2.13%	**0.01%**	1.63%	1.63%	0.20%	0.18%	1.11%	Trivial	
Bur26d	0.89%	2.12%	2.10%	**0.00%**	1.61%	1.61%	0.25%	0.12%	1.09%	Trivial	
Bur26e	0.89%	2.17%	2.11%	**0.01%**	1.61%	1.61%	0.16%	0.10%	1.08%	Trivial	
Bur26f	0.92%	1.88%	1.92%	**0.01%**	1.47%	1.47%	0.26%	0.24%	1.02%	Trivial	
Bur26g	0.96%	2.43%	2.36%	**0.00%**	1.76%	1.76%	0.31%	0.22%	1.23%	Trivial	
Bur26h	0.90%	1.96%	1.97%	**0.01%**	1.52%	1.52%	0.19%	0.18%	1.03%	Trivial	
Kra30a	17.41%	25.69%	25.89%	**3.13%**	22.72%	22.72%	4.35%	3.58%	15.69%	Moderate	
Kra30b	14.58%	23.98%	24.42%	**1.88%**	21.57%	21.57%	2.73%	1.89%	14.08%	Moderate	
Ste36a	54.10%	58.83%	60.19%	5.70%	53.91%	53.91%	4.72%	**4.46%**	36.98%	Hard	
Ste36b	89.97%	150.78%	154.68%	**8.22%**	130.96%	130.96%	11.22%	9.27%	85.76%	Hard	

Table 3.14 Eight Tuned Metaheuristics and Class IV Problem Instances

Name	GA-PM	GA-RK	IA-RK	IA-PM	SA-RK	SA-PM	TS-RK	TS-PM	AVG	Category According to Table 3.6	Category According to Stutzle (2006)
Tai20b	5.54%	9.05%	9.08%	**0.06%**	7.21%	7.21%	14.14%	14.72%	8.38%	Moderate	**Class IV**
Tai25b	15.12%	26.43%	25.03%	**0.27%**	17.84%	17.84%	16.18%	9.12%	15.98%	Moderate	
Tai30b	13.78%	22.67%	23.51%	**0.41%**	17.50%	17.50%	9.88%	12.46%	14.71%	Moderate	
Tai35b	13.00%	27.16%	26.12%	**0.98%**	21.51%	21.51%	7.00%	6.56%	15.48%	Moderate	
Tai40b	23.05%	34.87%	34.55%	**1.60%**	29.89%	29.89%	8.21%	6.57%	21.08%	Moderate	
Tai50b	20.80%	33.80%	33.37%	**2.57%**	30.19%	30.19%	5.53%	4.20%	20.08%	Moderate	
Tai60b	21.49%	35.67%	35.89%	**3.70%**	33.37%	33.37%	6.30%	4.20%	21.75%	Moderate	
Tai80b	22.91%	33.74%	34.21%	6.95%	32.44%	32.44%	4.20%	**3.49%**	21.30%	Moderate	
Tai100b	9.48%	11.25%	11.28%	5.52%	11.04%	11.04%	**2.36%**	3.52%	8.19%	Moderate	

The best percentage deviation values from the best-known solutions are boldfaced in each table. It should be noted that out of the eight methods, IA-PM, TS-RK, and TS-PM have shown promising results as opposed to the GAs and SAs.

Table 3.15 shows the comparison of the aforementioned three state-of-the-art methods and the best tuned methods from Tables 3.11-3.14 for each problem class. In the Table 3.15, each tuned metaheuristic (IA-PM, TS-RK, and TS-PM) with the minimum percentage of deviation from the best-known values are presented. These tuned metaheuristics yielded very promising results when compared to other state-of-the-art methods that have in-built greedy adaptive diversification and intensification mechanisms.

Table 3.15 Comparison of the Best Tuned Metaheuristics with State-of-the-Art Methods
(LS: Iterated Local Search, Ro-TS: Robust Tabu Search and MMAS: Ant Systems)
(T: Trivial Problems for Metaheuristics, M: Moderately Difficult Problems for Metaheuristics, and H: Hard Problems for Metaheuristics according to Table 3.6)

Class I					**Class II**					**Class III**					**Class IV**				
	Tuned (MIN)	LS	Ro-TS	MMAS		Tuned (MIN)	LS	Ro-TS	MMAS		Tuned (MIN)	LS	Ro-TS	MMAS		Tuned (MIN)	LS	Ro-TS	MMAS
Tai20a (M)	**0.00 (IA-PM)**	0.344	0.108	0.428	Nug30 (M)	0.070 (TS-PM)	**0.007**	0.013	0.042	Bur26a (T)	**0.00 (IA-PM)**	**0.00**	0.002	**0.00**	Tai20b (M)	**0.00 (IA-PM)**	**0.00**	**0.00**	**0.00**
Tai25a (M)	0.780 (TS-PM)	0.656	**0.274**	1.751	Sko42 (M)	0.470 (TS-PM)	**0.000**	0.025	0.104	Bur26b (T)	**0.00 (IA-PM)**	**0.00**	0.002	**0.00**	Tai25b (M)	**0.00 (IA-PM)**	**0.00**	**0.00**	**0.00**
Tai30 (M)a	0.860 (TS-PM)	0.668	**0.426**	1.286	Sko49 (M)	0.740 (TS-PM)	**0.068**	0.076	0.150	Bur26c (T)	**0.00 (IA-PM)**	**0.00**	0.002	**0.00**	Tai30b (M)	0.141 (IA-PM)	**0.00**	0.107	**0.00**
Tai35 (M)a	1.251 (TS-RK)	0.901	**0.426**	1.568	Sko56 (M)	0.640 (TS-PM)	**0.071**	0.088	0118	Bur26d (T)	**0.00 (IA-PM)**	**0.00**	0.002	**0.00**	Tai35b (M)	0.307 (IA-PM)	**0.00**	0.064	**0.00**
Tai40a (M)	1.802 (TS-RK)	1.082	**0.589**	1.131	Sko64 (M)	0.610 (TS-PM)	**0.057**	0.071	0.243	Bur26e (T)	**0.00 (IA-PM)**	**0.00**	0.002	**0.00**	Tai40b (M)	0.397 (IA-PM)	**0.00**	0.531	**0.00**
Tai50a (M)	2.047 (TS-RK)	1.211	**0.990**	1.900	Sko72 (M)	0.810 (TS-PM)	**0.085**	0.146	0.243	Bur26f (T)	**0.00 (IA-PM)**	**0.00**	0.002	**0.00**	Tai50b (M)	1.008 (IA-PM)	0.033	0.342	**0.002**
Tai60a (M)	1.964 (TS-RK)	1.305	**1.125**	2.484	Sko81 (M)	0.540 (TS-PM)	**0.082**	0.136	0.223	Bur26g (T)	**0.00 (IA-PM)**	**0.00**	0.002	**0.00**	Tai60b (M)	1.423 (IA-PM)	**0.00**	0.417	0.005
Tai80a (M)	1.802 (TS-RK)	1.029	**0.900**	2.103	Sko90 (M)	0.690 (TS-PM)	**0.128**	**0.128**	0.288	Bur26h (T)	**0.00 (IA-PM)**	**0.00**	0.002	**0.00**	Tai80b (M)	0.810 (TS-PM)	0.383	1.031	**0.096**
					Sko100a(M)	0.650 (TS-PM)	**0.109**	0.128	0.191	Kra30a (M)	0.903 (IA-PM)	**0.00**	0.268	0.314	Tai100b(M)	1.654 (TS-RK)	**0.083**	0.512	0.142
										Kra30b (M)	0.175 (IA-PM)	**0.008**	0.023	0.049					
										Ste36a (H)	2.687 (TS-PM)	**0.015**	0.155	0.181					
										Ste36b (H)	3.280 (IA-PM)	**0.00**	0.081	**0.00**					

In order to investigate the effects of the random-keys representations on the performance of GAs, we compared the greedy genetic algorithm developed by Ahuja *et al.* [3] with the tuned GA-RK. Ahuja *et al.* [3] compared 11 problem instances from the QAPLIB (Kra30b, Lipa50a, Lipa60a, Lipa70a, Lipa80a, Sko49, Sko90, Sko100a, Tho30, Wil50, and Wil100) and reported that the average deviation from the best known value is 11%. Table 3.16 and Figure 3.21 depict the comparison of the results of the tuned GA-RK with the greedy genetic algorithm with the nine problem instances from this study (Wil50 and Wil100 are not considered for this study).

Table 3.16 Comparison of Results of the Tuned GA-RK with the Greedy Genetic Algorithm

	Problem	Percentage Deviation from the Best-Known Value	
		Greedy Genetic Algorithm [3]	Tuned Random-Keys Genetic Algorithm (GA-RK)
1	Kra30b	30.00%	**20.35%**
2	Lipa50a	02.50%	**02.21%**
3	Lipa60a	02.50%	**02.00%**
4	Lipa70a	02.45%	**01.82%**
5	Lipa80a	02.40%	**01.63%**
6	Sko49	17.60%	**13.99%**
7	Sko90	15.10%	**13.31%**
8	Sko100a	14.90%	**11.89%**
9	Tho30	25.00%	**16.89%**
Average Deviation %		**12.49%**	**09.34%**

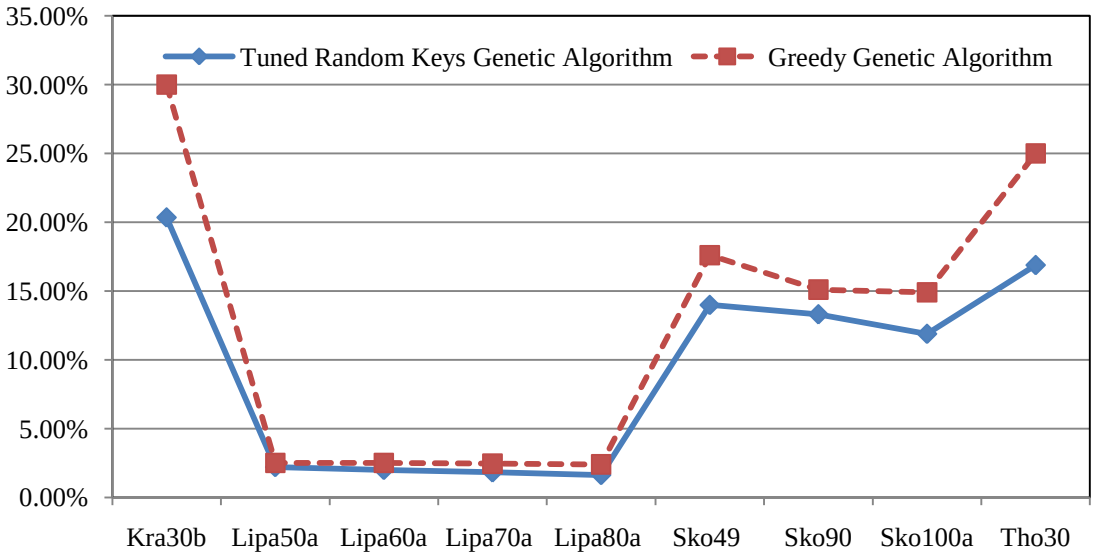


Figure 3.21: Tuned GA-RK with the Greedy Genetic Algorithm

3.12 Conclusions

In this chapter, four metaheuristics were investigated using two representation schemes on the QAP. The extensive parameter-tuning process significantly improves the performance of any given metaheuristic. However, it was found that the representation scheme plays an important role as well. In the case of IA, this was particularly evident.

Statistical analysis concluded that path-based metaheuristics, such as SA and TS, are more robust to representation schemes than population-based metaheuristics. IA was significantly improved once the permutation representation was introduced. Even though random-keys representation became competitive in other combinatorial optimization problems, such as scheduling, the permutation representation is more suitable for cases like QAPs.

Based on the results of the parameter-tuning procedure, QAPLIB problems were categorized into three phases, including trivial problems, moderately hard problems, and hard problems. For further analysis in later chapters, the moderate and hard problems were only considered, while the trivial problems were discarded.

In order to carry out a firm comparison study, we utilized the four classes of problems from the literature and compared the results of the tuned metaheuristics with these classes in addition to comparing the solution quality with the three state-of-the-art metaheuristics. Investigating the parameter-tuning process and the representation scheme played an important role in determining solution quality and yielded promising results compatible with the integration of more sophisticated diversification and intensification mechanisms. Overall, TS with both representation schemes and IA with permutation representation seemed to outperform the rest of the metaheuristics investigated here. Finally, we compared the performance of our tuned random-keys GA with the greedy GA initially developed by Ahuja *et al.* [3]. The underlying results of this comparison indicated that the tuned GA outperformed the greedy GA for the selected problem instances.

CHAPTER FOUR

4. COMPARISON STUDY: SOLUTION REPRESENTATION

4.1 Introduction

In this chapter, we investigate the performance of each tuned path-based and population-based metaheuristic using measures pertaining to problem characteristics. We use problem size, flow and distance dominance measures, sparsity (number of zero entries in the matrices), and the coefficient of correlation measures of the matrices to build search trajectories.

The broad objectives of this chapter include the following:

- Classify the selected problem instances of the QAPLIB based on the aforementioned measures using classification and regression trees (CART)
- Identify the determinants of a difficult problem instance
- Identify the characteristics that are common (unique) to each path-based and population-based method
- Identify pillars to build a unified framework for the selected metaheuristics to aid future research on the QAP

In this chapter we integrate the findings of chapter 3 in which we extensively tuned four metaheuristics using two representation schemes. In this chapter, we begin with the eight tuned metaheuristics and perform an overall comparison study to visualize the spread of the data. This is presented via a series of box-plots representing each metaheuristic method with respect to each representation scheme.

The rest of this chapter unfolds as follows: In section 2, a detailed comparison study is carried out with measures on problem characteristics. These measures include

the aforementioned size of the problem, flow and distance dominance, sparsity of the matrices, and correlation between the flow and distance matrices. Section 3 discusses the results obtained from section 2, and the chapter concludes with a chapter summary and conclusions to lay the foundation for chapter 5.

4.2 Initial Comparison

We begin by comparing each path-based metaheuristics (SA and TS) and population-based metaheuristic (GA and IA) developed using random keys. These four tuned methods were statistically evaluated to identify any significant characteristics pertaining to philosophical differences. First of all, the average loss function values of the four random-keys-based metaheuristics are depicted in the box plots of Figures 4.1 and 4.2. For statistical analysis, Friedman's test was used utilizing the MiniTab® and R® statistical packages. The Friedman's test results indicate that TS-RK outperformed SA-RK ($p < 0.001$). In Figure 4.1, it is clearly noticeable that the spread of the data is very narrow in TS-RK whereas, for SA-RK, the spread is very wide. In Figure 4.2, the population-based comparison is shown with GA-RK and IA-RK. However, the visualization from this particular graph does not provide enough evidence to select a winning method out of the two. Furthermore, we were unable to conclude if there was any statistical difference between the two population-based algorithms (GA-RK and IA-RK).

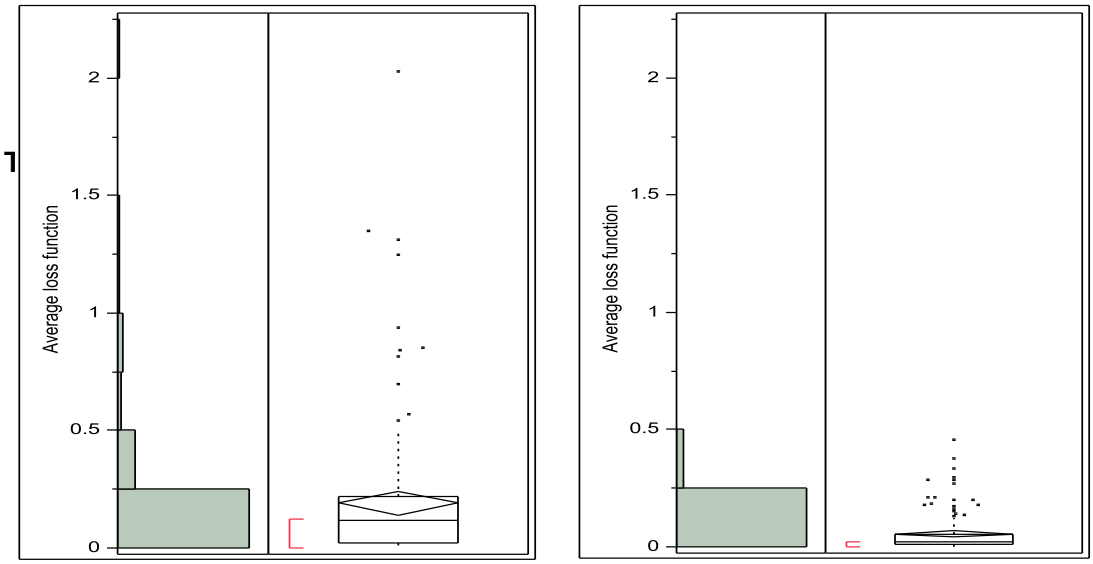


Figure 4.1: Box Plots of the Path-Based Random Keys—SA-RK (left) and TS-RK (right)

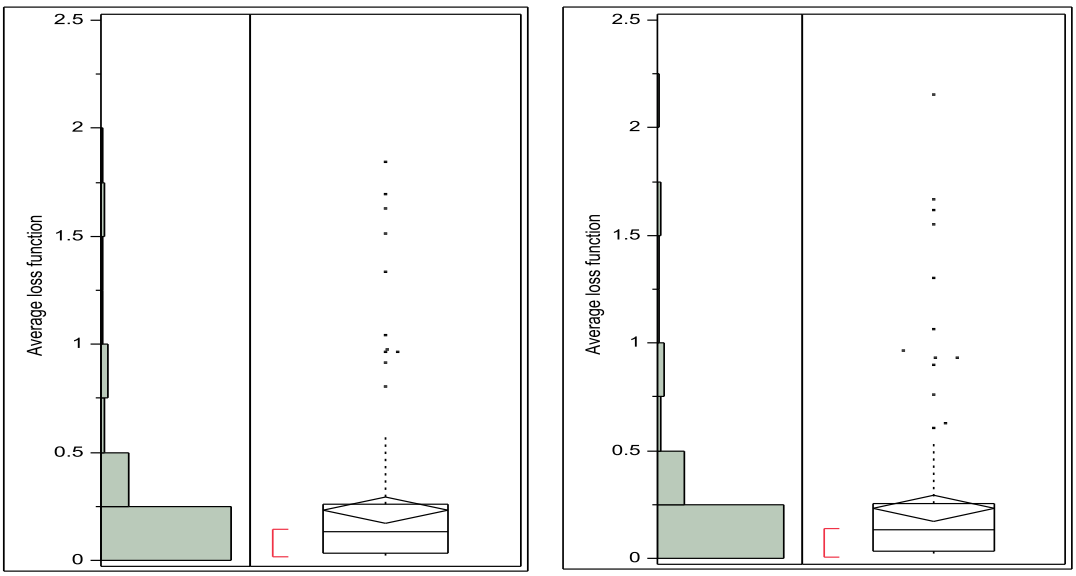


Figure 4.2: Box Plots of the Population-Based Random Keys—GA-RK (left) and IA-RK (right)

Next, the average deviations of the permutation representations were compared. Figures 4.3 and 4.4 depict the box-plot visualizations of the four methods. For statistical analysis, the Friedman's test results indicate that TS-PM outperformed

SA-PM ($p < 0.001$) in path-based analysis and that for population-based analysis, IA-PM outperformed GA-PM ($p < 0.001$).

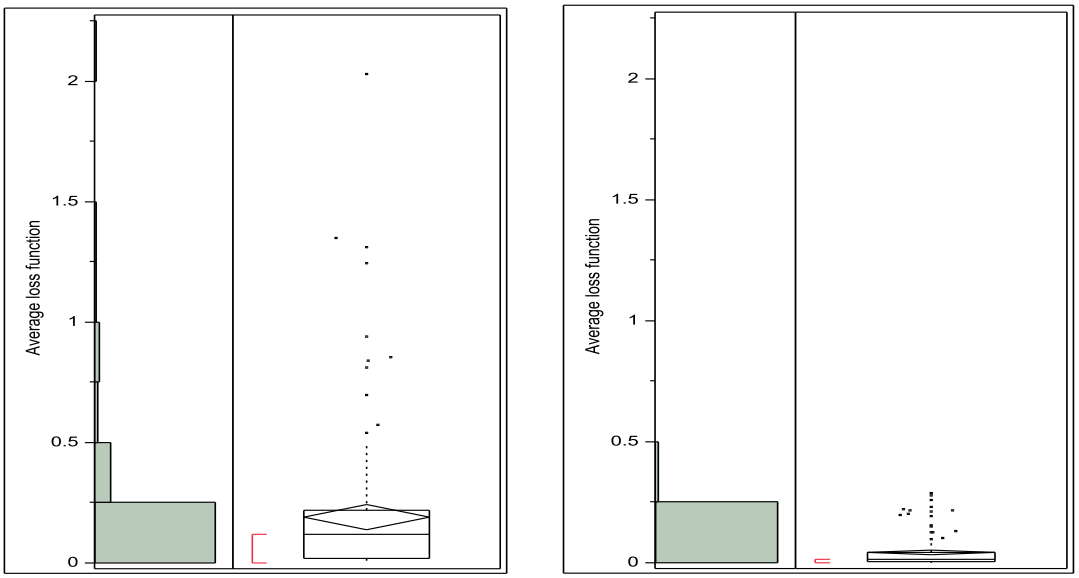


Figure 4.3: Box Plots of the Path-Based Permutations—SA-PM (left) and TS-PM (right)

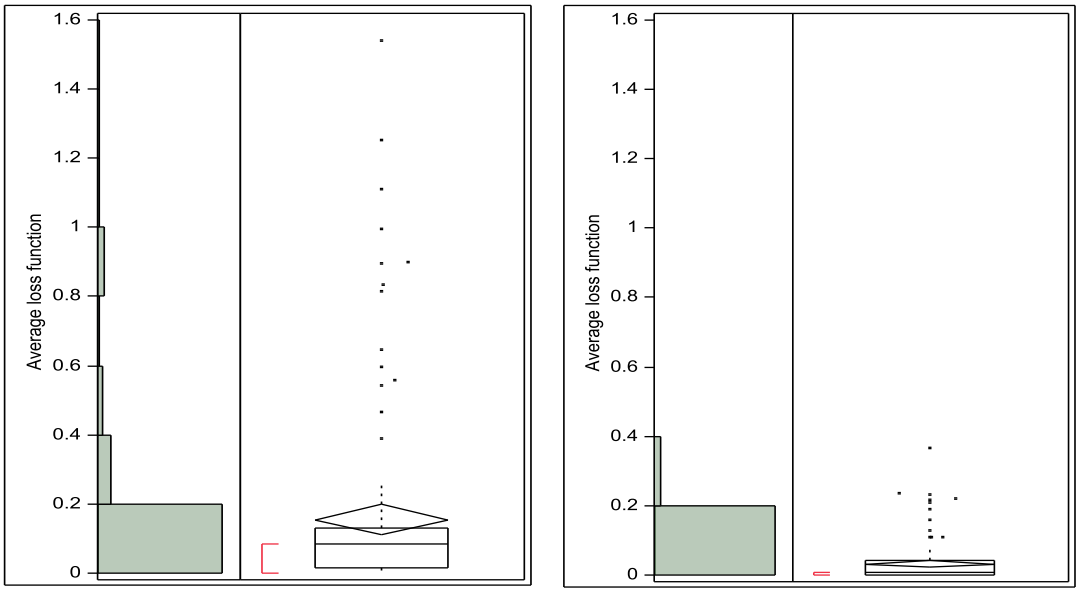


Figure 4.4: Box Plots of the Population-Based Permutations—GA-PM (left) and IA-PM (right)

Unlike the random keys, the permutation representation tends to show a significant difference in terms of improving the solution quality. Specifically, for

population-based methods, a drastic improvement in solution quality was observed. The initial comparison study concludes with the identification of the best metaheuristic for each path-based and population-based method. For example, out of SA-RK and TS-RK, TS-RK was selected since it yielded the fewest average loss-function values. In the case of GA-RK and IA-RK, GA-RK was selected even though there was no statistically significant difference between the two. However, there were fewer outliers in GA-RK. The following Figure 4.5 depicts the box plots of the four methods. When all four methods were compared, IA-PM outperformed the rest ($p < 0.001$).

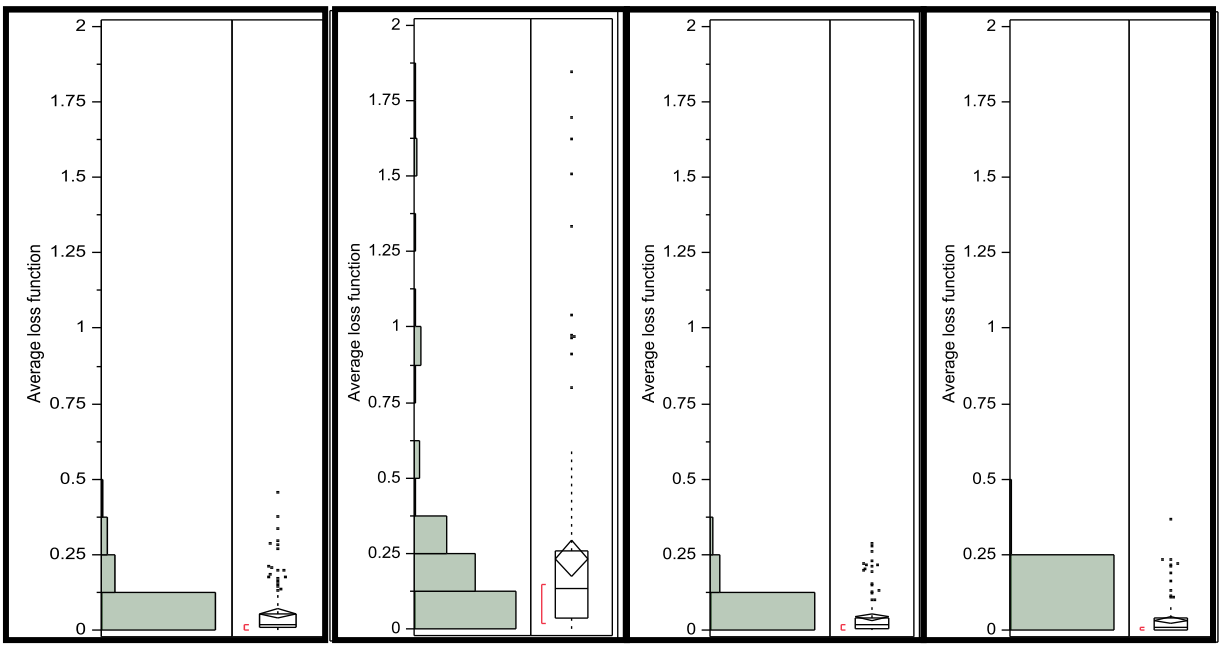


Figure 4.5: Overall Comparison of TS-RK, GA-RK, TS-PM, and IA-PM

In the next section, the metaheuristics are further analyzed using different performance measures. This analysis will be used for identifying the detailed characteristics of each method and how each metaheuristic's performance is affected by attributes pertaining to data matrices.

4.3 Detailed Comparison

In this section, the performance of the eight tuned metaheuristics (GA-RK, GA-PM, IA-RK, IA-PM, SA-RK, SA-PM, TS-RK, and TS-PM) is analyzed using the problem characteristics mentioned before. For this analysis, we will use the hard problem instances discussed in chapter 3 (Table 3.6).

This detailed analysis is presented in the form of three separate studies with a path-based comparison, a population-based comparison, and an overall comparison for the 14 input files from the QAPLIB. These problem instances were found to be very difficult to solve with the extensive parameter-tuning process and were presented in the problem classification in chapter 3 and are presented here in Table 4.1.

Table 4.1: Hard Problems for Parameter Tuning (14 Files)

File Names	Problem Size	Flow Dominance (fd)	Distance Dominance (dd)	Sparsity	Coefficient of Correlation
Chr15a	15	69.8909	327.68	0.808	0.163783
Chr15b	15	69.8909	327.68	0.808	0.029836
Chr15c	15	69.8909	327.68	0.808	0.012951
Chr18a	18	63.1960	351.13	0.839	-0.006276
Chr20a	20	59.4589	346.37	0.855	0.014580
Chr20b	20	59.4589	346.37	0.855	-0.171070
Chr20c	20	65.7126	346.37	0.855	0.088583
Chr22c	22	57.9713	424.26	0.883	0.014315
Esc32a	32	69.2714	281.56	0.667	-0.063071
Esc32b	32	69.2714	208.26	0.601	-0.190368
Esc128	128	52.0396	1153.90	0.929	0.258538
Ste36a	36	400.305	55.64	0.706	-0.077467
Ste36b	36	400.305	100.79	0.706	-0.077467
Ste36c	36	400.305	55.90	0.706	-0.107290

Using this detailed comparison study; we intend to compare the solutions generated by the eight methods using the aforementioned input file characteristics.

This process enables us to generate insights as to why these 14 input files were identified as difficult problems to solve.

4.4 Performance Measures

The performance measures of the comparison study include the size of the problem, flow and distance measures, sparsity of the matrices, and correlation of the flow and distance dominance measures. The problem sizes range from 15 to 128 and represent Chr*, Esc*, and Ste* problem instances of the QAPLIB. The flow and distance dominance measures (fd, dd) were computed for the 14 files using the following mathematical equations (1 and 2).

$$f(\text{Distance or Flow}) = 100\frac{\sigma}{\mu} \tag{1}$$

$$\mu = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij} \quad \text{and} \quad \sigma = \sqrt{\frac{1}{n^2-1}\sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij} - \mu^{2}} \tag{2}$$

According to Stützle, (2006) [83], the sparsity of the flow and distance matrices plays an important role and can be defined by the number of zero elements in each matrix. The following formula was used to compute the sparsity measures (S_p) for each input file (3 and 4).

$$S_p(\text{Sparsity of matrices}) = \frac{n_0}{n^2} \tag{3}$$

n_0 (number of zero elements), n (size of the matrix)

$$S_p(\text{Sparsity}) = \left| S_p(\textit{flow matrix}) - S_p(\text{distance } \textit{matrix}) \right| \tag{4}$$

As another measure to compare the characteristics of the input data, we use the coefficient of correlation (r^2) between the two matrices. This enables us to analyze

the data matrices and the possible correlation measures between the two data matrices. The following formula was used for the correlation analysis.

$$r^2 = \frac{(\sum xy - n\bar{x}\bar{y})^2}{(\sum x^2 - n\bar{x}^2)(\sum y^2 - n\bar{y}^2)} \qquad (5)$$

The analysis is presented in two separate sections depending on the two broad categories of the metaheuristics: namely, the path-based comparison and the population-based comparison. In a later section, we compare the two categories and present the findings.

4.4.1 Path-based Comparison

We began the path-based comparison study by analyzing the problems size. Figure 4.6 depicts how each path-based metaheuristic method performed with respect to its problem size. It is clearly noticeable that the performance of SA-RK and SA-PM are directly affected by the size of the problem, which is not the case, however, for TS-RK and TS-PM.

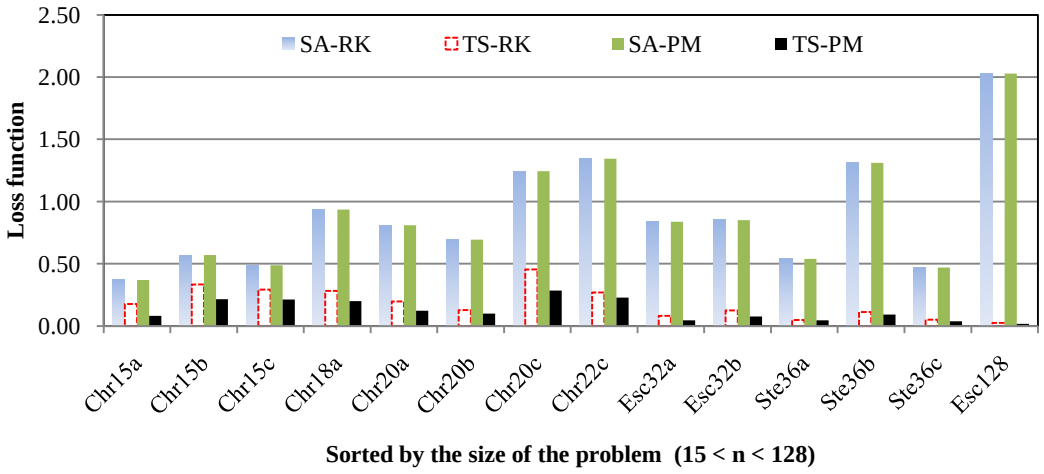


Figure 4.6: Performance of the Path-Based Metaheuristics with Problem Size

Figure 4.7 and 4.8 present the flow and distance dominance measures for the four methods. The input files selected show either a very high dominance measure or

a very low dominance measure, which can be observed for both measures. It should be noted that TS-RK and TS-PM are quite robust for flow and distance dominance measures as opposed to SA-RK and SA-PM. Specifically, the latter two show erratic behaviors for the two dominance measures.

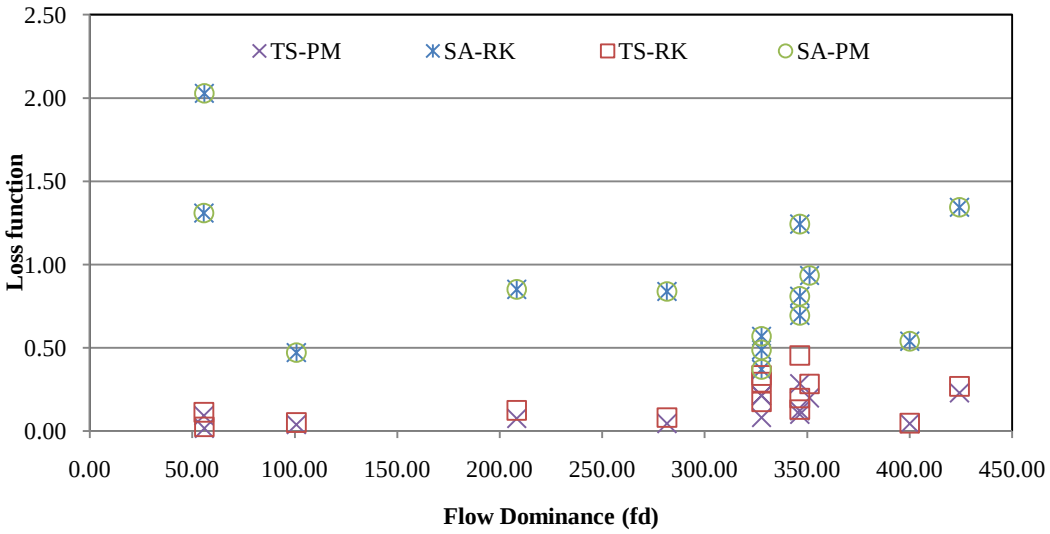


Figure 4.7: Performance of the Path-Based Metaheuristics with Flow Dominance (fd)

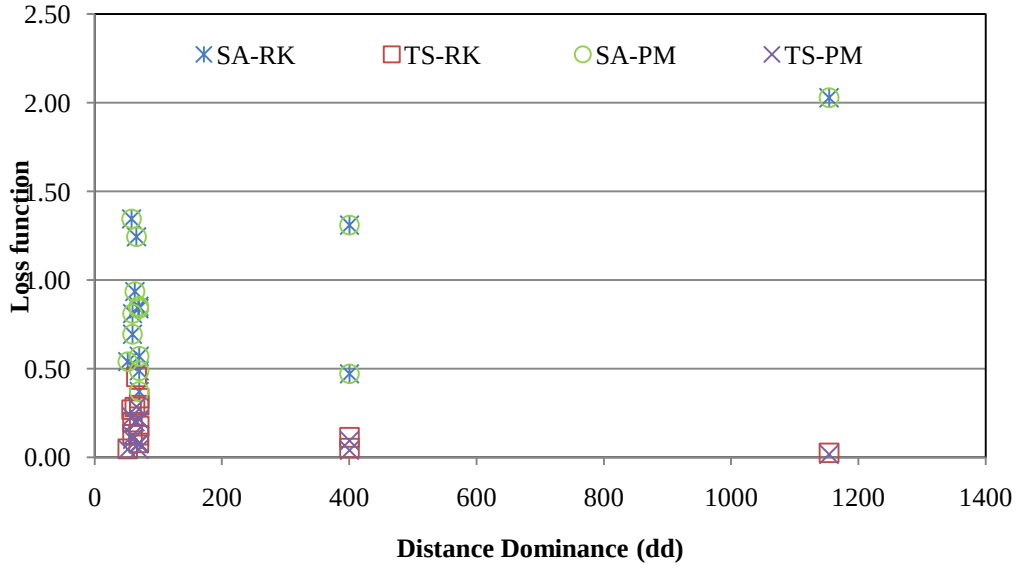


Figure 4.8: Performance of the Path-Based Metaheuristics with Distance Dominance (dd)

Figure 4.9 depicts the sparsity measures for the 14 input files. Generally, higher sparsity values indicate that the problems are based on real-world problems, while lower values indicate that the problems are pseudo randomly generated [83]. It

is clearly visible that when the sparsity values are high, SA-RK and SA-PM tend to show inferior results. The same phenomenon is applicable to TS-RK and TS-PM; however the effect is less significant.

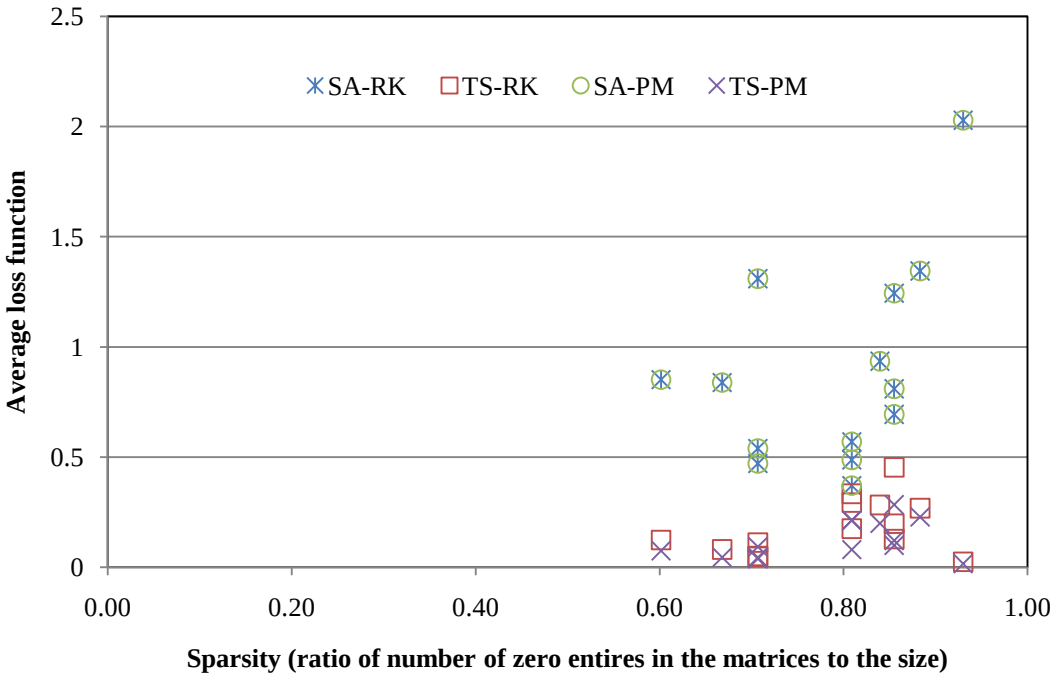


Figure 4.9: Performance of the Path-Based Metaheuristics with Sparsity of Matrices

Figure 4.10 depicts the correlation measures for the 14 input files. For TS-RK and TS-PM, when the correlation coefficients are positive, the performance tends to degrade. For SA-RK and SA-PM, we cannot indentify a specific pattern. However, when the correlation is high, the performance can be adversely affected.

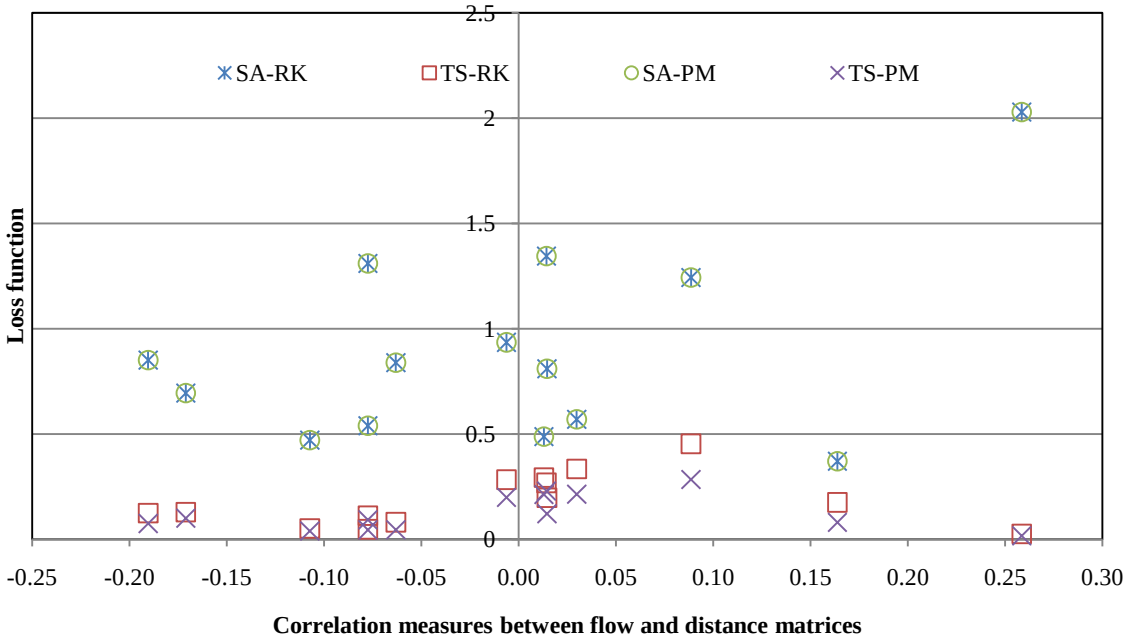


Figure 4.10: Performance of the Path-Based Metaheuristics with Correlation of the Matrices

As we summarize the path-based comparison, it is clear that the TS is more robust to all of the problem characteristics than the SA. TS is well suited for QAP-like problem instances irrespective of the domain (randomly generated or reality based).

4..2 Population-based Comparison

The population-based comparison study begins with problem size. Figure 4.11 depicts how each population-based metaheuristic method performs with respect to problem size. The performance of GA-RK, GA-PM, and IA-RK are directly affected by the size of the problem as opposed to IA-PM. There is a significant difference in the performance of IA with respect to the two different representation schemes. IA-PM does much better than the rest of the population-based methods irrespective of the size of the problem.

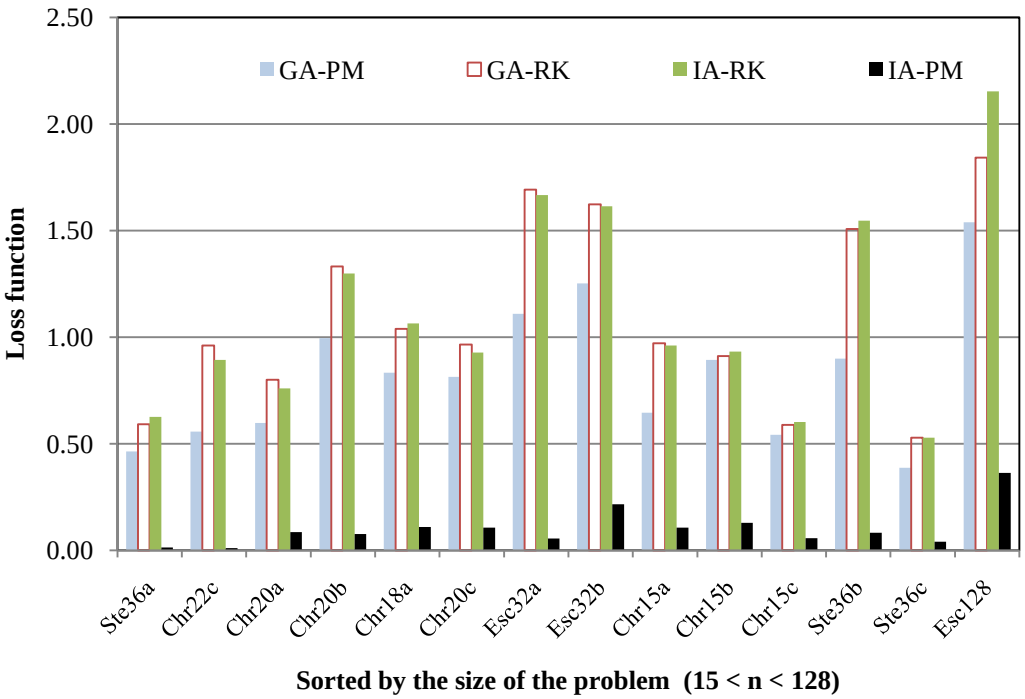


Figure 4.11: Performance of the Population-Based Metaheuristics with Problem Size

Figure 4.12 and 4.13 present the flow and distance dominance measures for the four population-based methods. From the figures, it is evident that higher flow/distance dominance measures create problems for the GAs and IA-RK methods. However, IA-PM is quite robust to these measures as opposed to the rest.

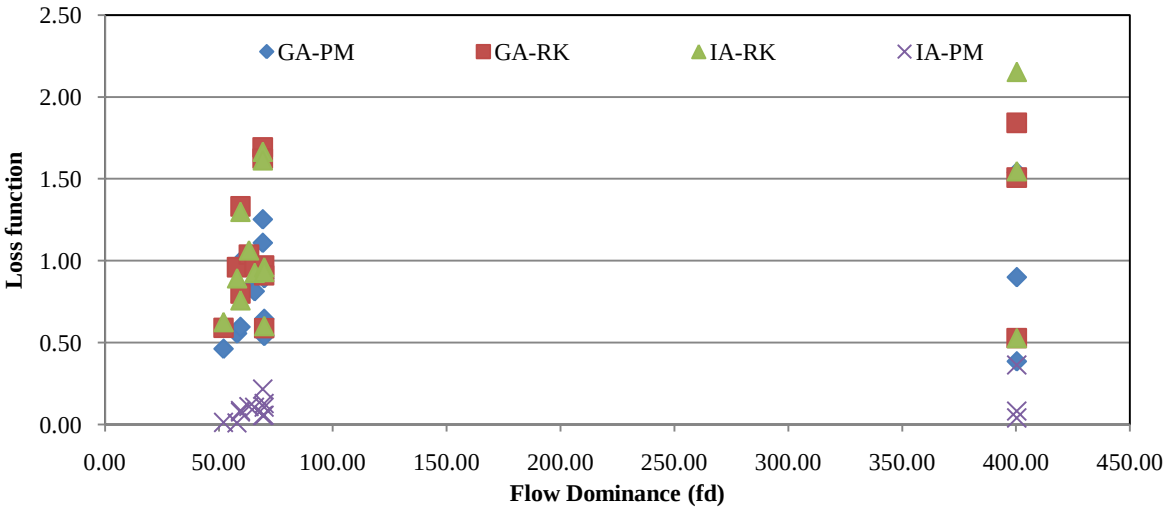


Figure 4.12: Performance of the Population-Based Metaheuristics with Flow Dominance (fd)

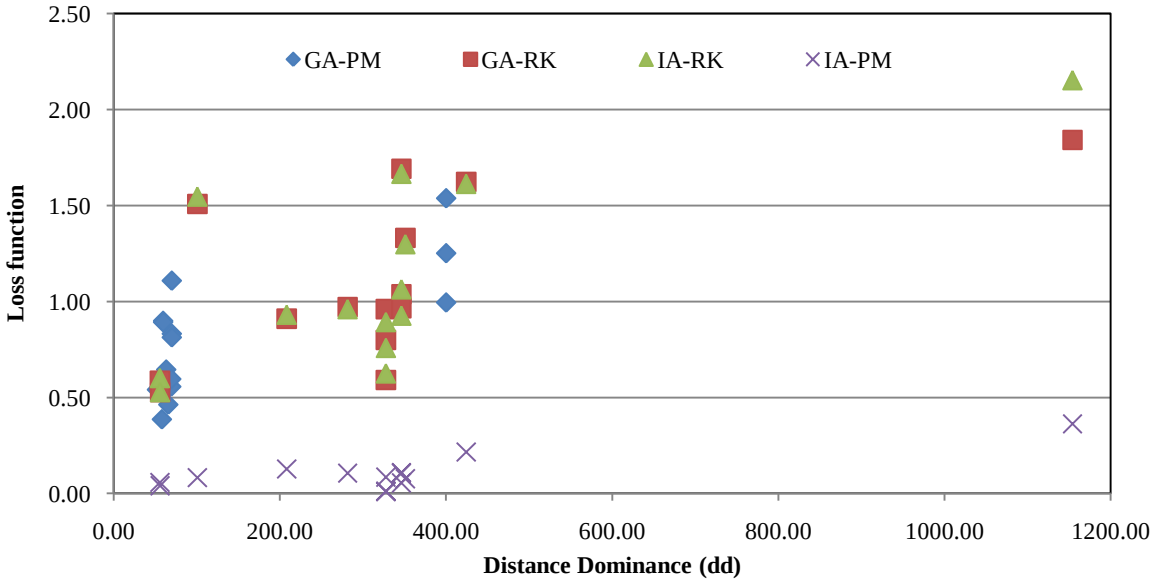


Figure 4.13: Performance of the Population-Based Metaheuristics with Distance Dominance (dd)

Figure 4.14 depicts the performance vs. sparsity measures for population-based methods. It is clearly visible that when the sparsity values are high, GAs and IA-RK tend to show inferior results. However, IA-PM is less affected by this measure.

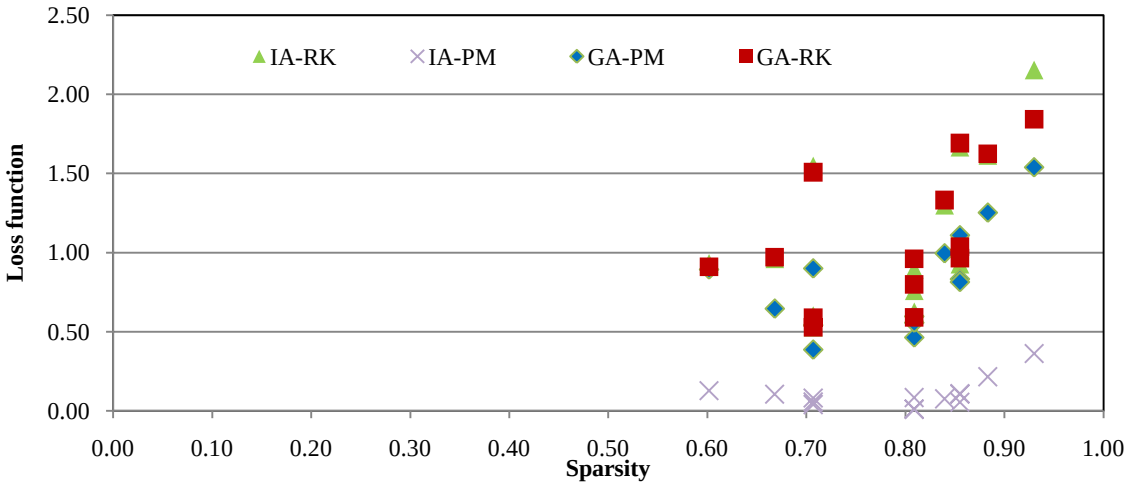


Figure 4.14: Performance of the Population-Based Metaheuristics with Sparsity of the Matrices

Figure 4.15 depicts the correlation measures, and a significant difference in performance can be seen for IA-PM. For the other methods, higher correlation causes the performance to degrade. However, when the correlation is high, the performance can be adversely affected. It is clearly noticeable that GA-RK and IA-RK show similar performance for each correlation value with GA-PM doing slightly better than GA-RK at the end.

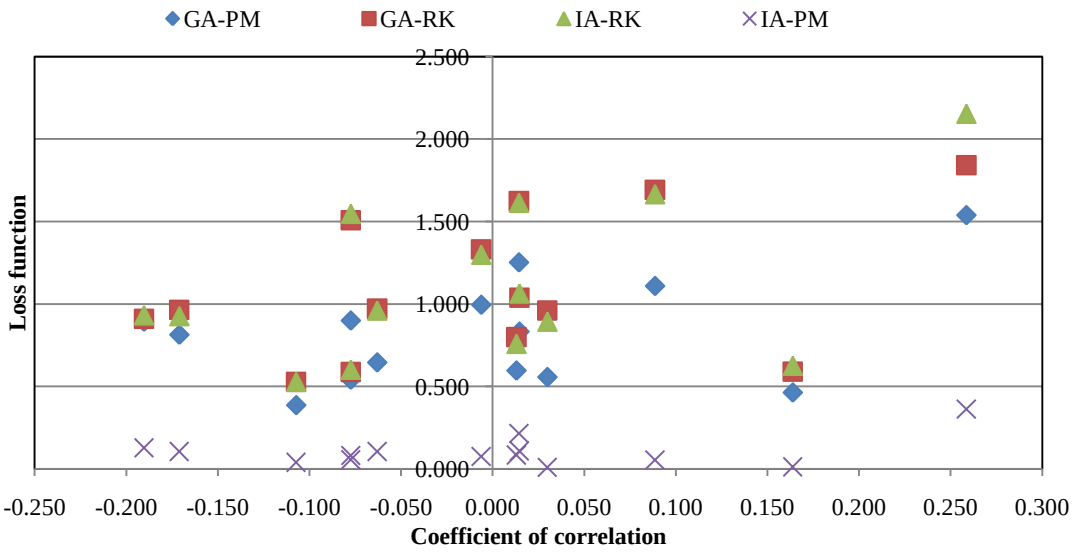


Figure 4.15: Performance of the Population-Based Metaheuristics with Correlation Measures

This concludes the population-based comparison. The next section analyzes the methods using Classification and Regression Trees (CART) tools. This next section is presented as an overall comparison to generate more insights.

4.7 Classification and Regression Trees (CART)

In this phase, we analyze the eight metaheuristics and the selected problem characteristics using CART tools and two approaches. In the first phase, we use all of the problem instances (130 files from the QAPLIB), while in the second phase, we use on the hard problems (discussed earlier in this chapter). The results of these two phases will be used to identify the determinants of trivial problem instances.

4.5.1 Overall Comparison- Phase I

For the CART analysis, there were two dependent variables, which we analyzed separately with the JMP® statistical package. The two variables included the type of algorithm and the loss-function values for each method, and the independent variables were problem size, flow dominance, distance dominance, sparsity, and coefficients of correlation. Figure 4.16 depicts the CART graph for the 130 input files with the type of algorithm as the dependent variable. According to figure 4.16, the first criterion for partition is the loss-function value by which the eight methods can be clearly separated by high (loss > 0.605) and low (loss < 0.605) values. The second splitting criterion is the distance dominance measure. Likewise, the partitioning process can be performed for as many as partitions we require. The key here is to identify the determinants of trivial and hard problems. According to this CART

analysis, the importance of each independent criterion can be ordered as follows: distance dominance, coefficient of correlation, problem size, sparsity, representation scheme, and flow dominance. When loss function is considered as the dependent variable, the following results are obtained and are also presented in Figure 4.17. From this analysis, the prominent criterion is sparsity followed by the distance dominance.

From the overall comparison of the 130 input files, it is clear that sparsity, distance dominance, and representation scheme are important determinants of performance as opposed to the rest.

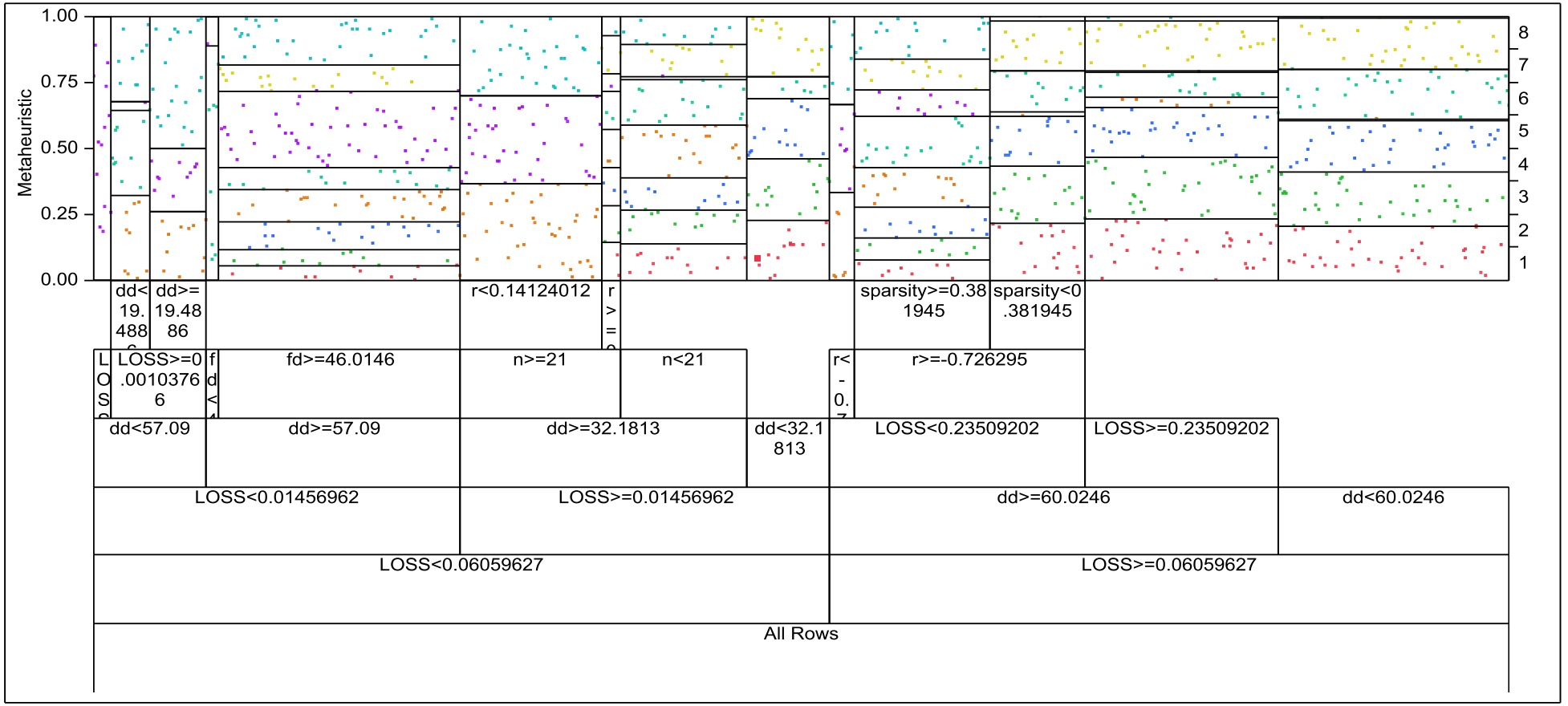


Figure 4.16: Partition by Metaheuristic Method for the 130 Input Files
(GARK-1, IARK-2, SARK-3, TSRK-4, GAPM-5, IAPM-6, SAPM-7, and TSPM-8**)**

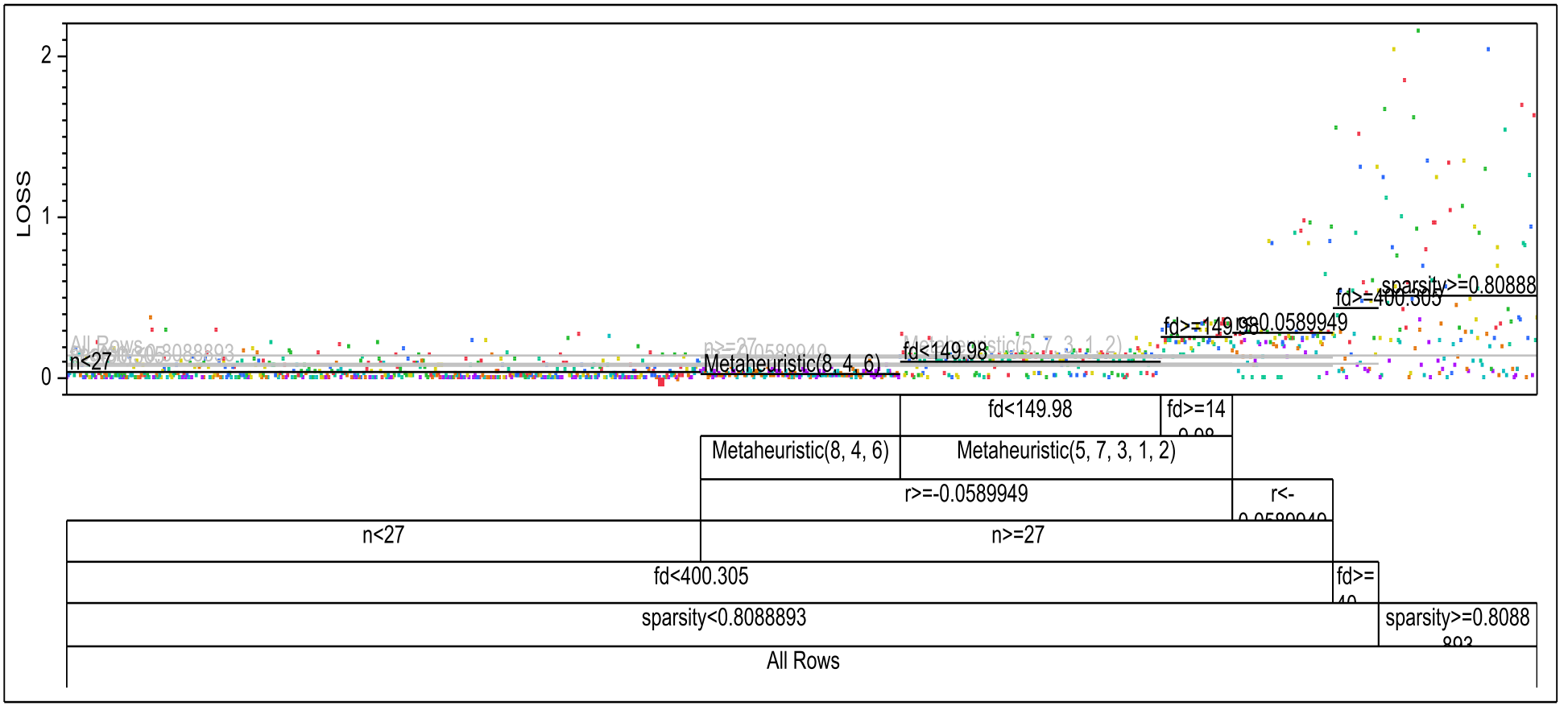


Figure 4.17: Partition by Loss Function for the 130 Input Files
(GARK-1, IARK-2, SARK-3, TSRK-4, GAPM-5, IAPM-6, SAPM-7, and TSPM-8**)**

In order to see how each method behaves, the CART analysis was carried out for each metaheuristic. Table 4.2 depicts each method with each criterion considered when the splits were made. These criteria were ordered in terms of their importance: criterion 1 being the most important and criterion 5 being the least important. It can be seen that for many methods, sparsity plays an important role in determining the performance, while distance dominance plays the least important role.

Table 4.2: CART Analysis for Each Metaheuristic for the 130 Files

Metaheuristic	Criteria for CART Analysis				
	Criterion 1 (1^{st} spilt)	Criterion 2 (2^{nd} spilt)	Criterion 3 (3^{rd} spilt)	Criterion 4 (4^{th} spilt)	Criterion 5 (5^{th} spilt)
GA-RK	**Sparsity**	Flow dominance	Problem size	Coefficient of Correlation	**Distance dominance**
GA-PM	Flow dominance	Coefficient of Correlation	Distance dominance	Sparsity	Problem size
IA-RK	**Sparsity**	Flow dominance	Problem size	Coefficient of Correlation	**Distance dominance**
IA-PM	Coefficient of Correlation	Sparsity	Problem size	Distance dominance	Flow dominance
SA-RK	**Sparsity**	Flow dominance	Problem size	Coefficient of Correlation	**Distance dominance**
SA-PM	**Sparsity**	Flow dominance	Problem size	Coefficient of Correlation	**Distance dominance**
TS-RK	**Sparsity**	Coefficient of Correlation	Flow dominance	Distance dominance	Problem size
TS-PM	Coefficient of Correlation	Sparsity	Problem size	Flow dominance	**Distance dominance**

4.5.2 Overall Comparison- Phase II

In the second phase, we use the hard (14 files total) instances from the 130 problems (discussed earlier in this chapter) and analyze them using the CART tools. Figure 4.18 depicts the CART graph for the 14 input files with the algorithm name as the dependent variable. According to figure 4.18, the first criterion for partition is the loss-function value; the eight methods can be clearly separated by high (loss > 0.370) and low (loss < 0.370) values. The second splitting criterion is the size of the problems followed by the sparsity and flow dominance measures. In Figure 4.19, we consider loss function as the dependent variable, and the most prominent criterion for partitioning is sparsity followed by the coefficient of correlation and flow dominance measures. One important observation is that we can clearly partition the space by looking at the metaheuristics method as well. In this case, there is a performance difference between the best performing metaheuristics; for example, look at the performance difference between metaheuristic number 4 (TSRK) and metaheuristic numbers 6 and 8 (IAPM and TSPM). In a later stage of the partition, the difference between metaheuristics 3, 5, and 7 (SARK, GAPM, and SAPM) are highlighted in contrast to 1 and 2 (GARK-1, and IARK).

From this analysis, as the importance of each criterion can be summarized as follows: sparsity, representation scheme, size of problem, coefficient of correlation, flow dominance, and distance dominance.

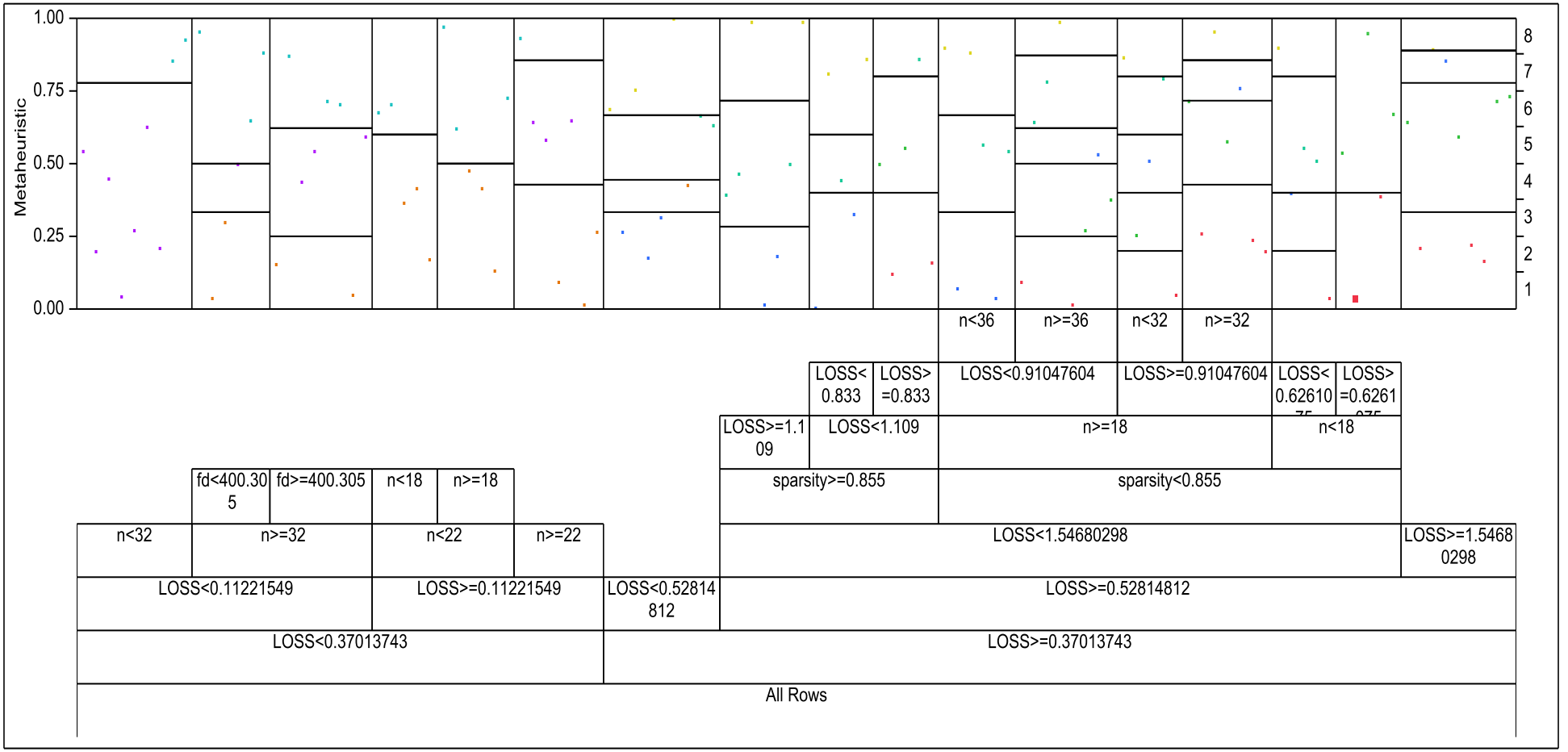


Figure 4.18: Partition by the Metaheuristics Method for the 14 Input Files (Hard Problems)
(GARK-1, IARK-2, SARK-3, TSRK-4, GAPM-5, IAPM-6, SAPM-7, and TSPM-8**)**

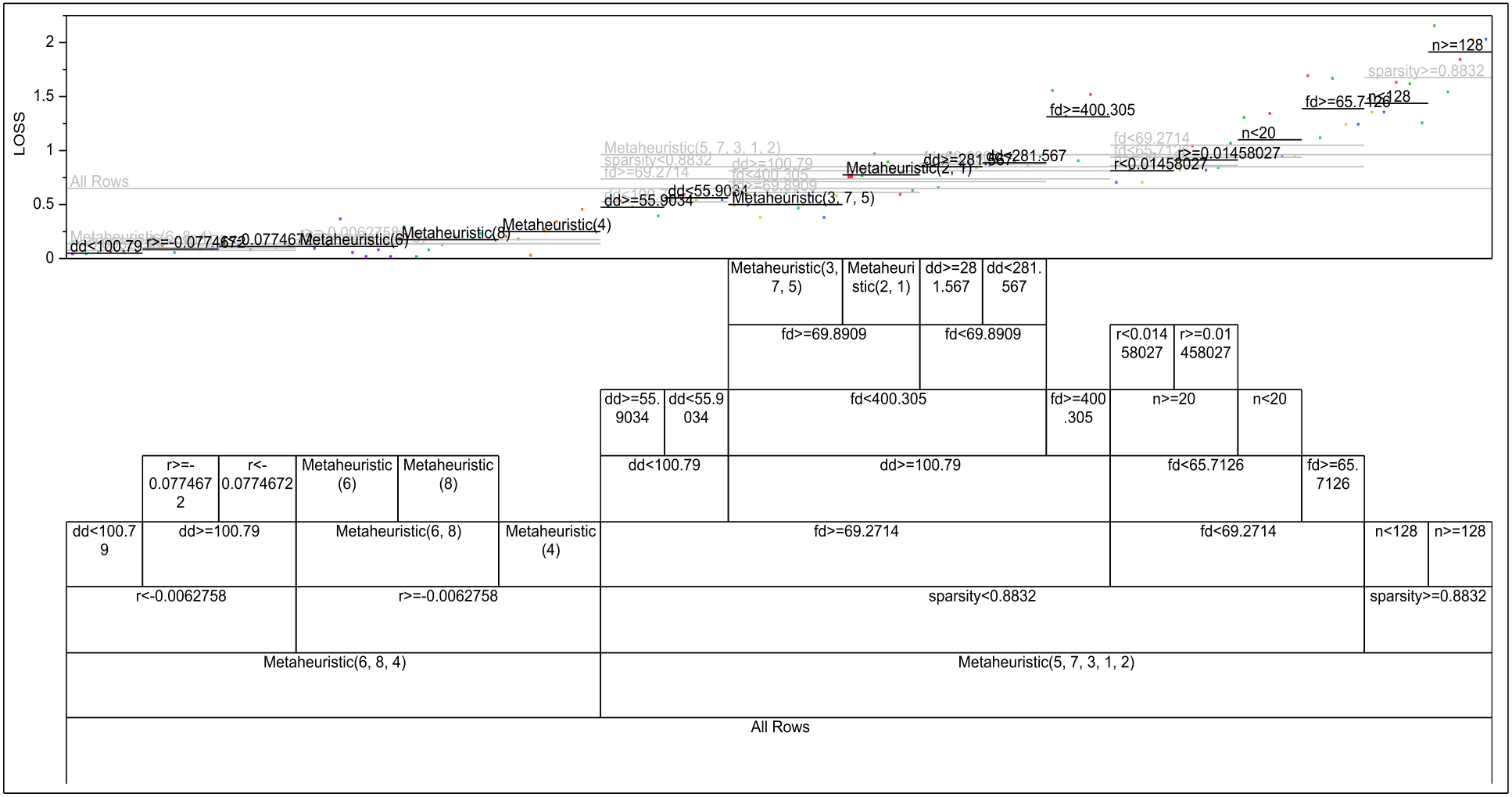


Figure 4.19: Partition by Loss Function for the 14 Input Files (Hard Problems)
(GARK-1, IARK-2, SARK-3, TSRK-4, GAPM-5, IAPM-6, SAPM-7, and TSPM-8**)**

The same individual analysis was carried out for each method for the 14 input files. Table 4.3 depicts each method and each criterion considered when the splits were made. It can clearly be seen that in contrast to Table 4.2, this analysis does not use the entire set of criteria, meaning that the splitting process ends with only one criterion. Also, it is evident that for the GAs and IAs (population-based methods), sparsity and flow dominance play an important role in determining performance. For the path-based methods, sparsity and the size of the problem determine the performance.

Table 4.3: CART Analysis for Each Metaheuristic for the 14 Files

Metaheuristic	Criteria for CART Analysis
	Criterion 1 (1^{st} spilt)
GA-RK	Flow dominance>=69.27
GA-PM	Sparsity>=0.839
IA-RK	Sparsity>=0.839
IA-PM	Flow dominance>=65.71
SA-RK	Sparsity>=0.855
SA-PM	Sparsity>=0.855
TS-RK	Problem size>=32
TS-PM	Problem size>=32

4.6 Overall Comparison- Findings

The aforementioned analyses raise the following questions:

- Are path-based metaheuristic approaches more suitable for QAPs?
- Are population-based metaheuristics not suitable for these problems?
- Are permutation representations more suitable than random keys representation schemes?
- What unique features govern the efficiency of an algorithm, and what do the search trajectories look like?

To explore solutions to the above questions, the next phase of this study examines the individual search paths of each algorithm. For this analysis, we use the 14 hard problems discussed before and track the individual search trajectories of each metaheuristic method.

The individual search trajectories of the eight methods were tracked using a different computer program with which every non-overlapping best solution was recorded. Each method used the same number of fitness evaluations to create a fair comparison. These 14 files are presented in one of three figures depending on the domain of the problem. Figure 4.20 shows the eight Chr* files, figure 4.21 shows the three Esc* files, and figure 4.22 depicts the three Ste* files. These graphs have been plotted using the value fitness function and the number of best solutions found per metaheuristic method. From Figures 4.20, 4.21, and 4.22, it can be seen that the path-based methods start with less attractive solutions than the population-based methods. However, as time passes, the path-based methods improve drastically as compared to the population-based methods in terms of the number of best solutions found and the overall quality of the solutions. This phenomenon is common to all the hard problems. In terms of final performance, at the end of each run, TS-RK, TS-PM, and IA-PM outperform the rest of the methods. When efficiency is taken into consideration, IA-PM does better than the two TSs. where the number of good solutions were found with less exploration of the search space, IA-PM was able to find the best known solution most of the time whereas, for the two TSs, the best known solutions were found after a full exploration of the search space.

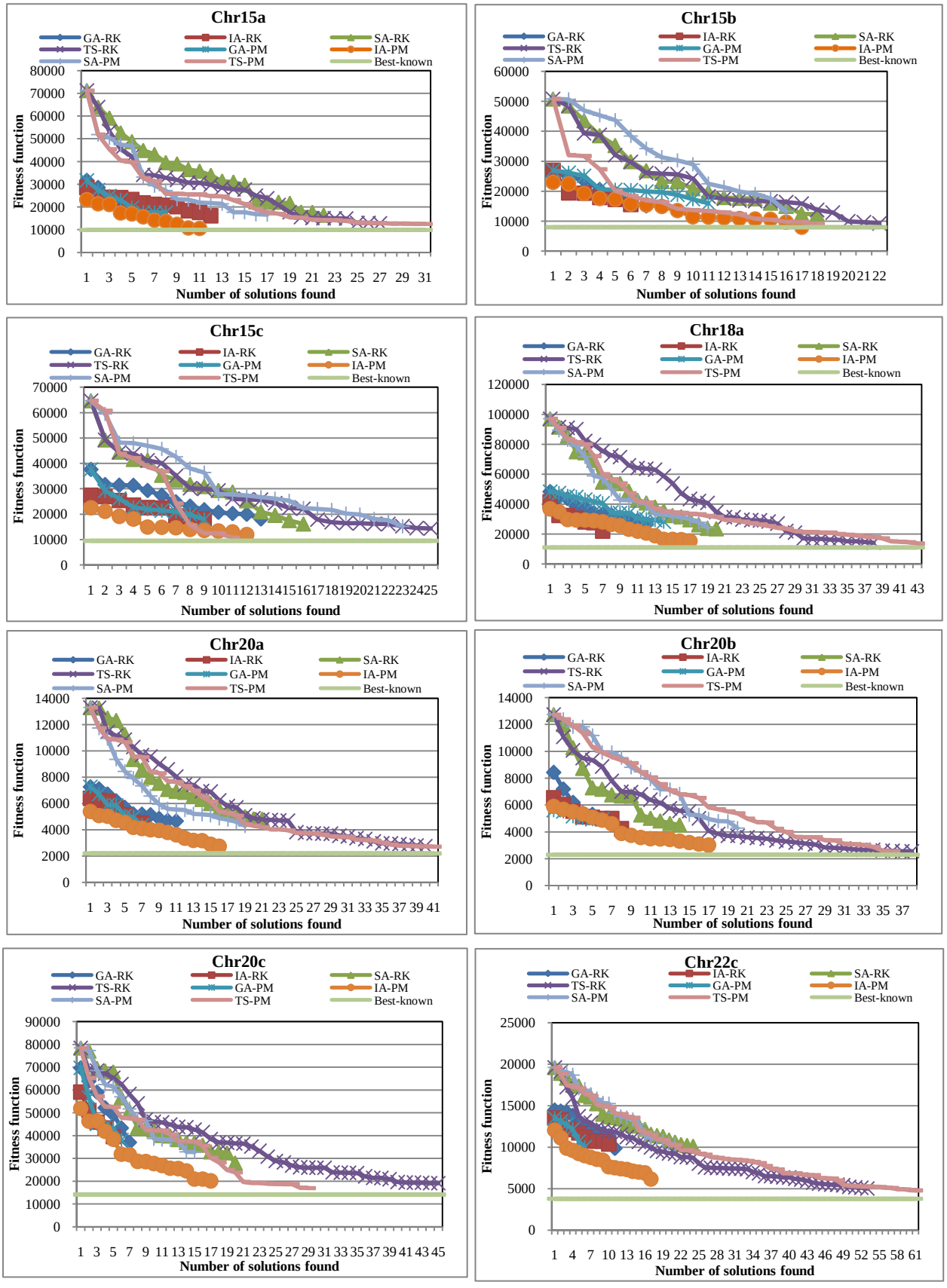
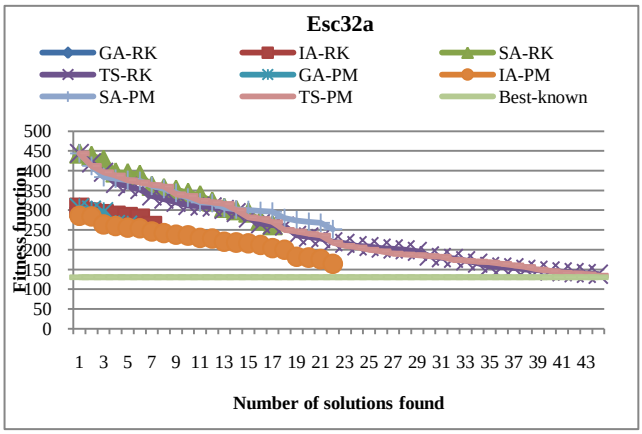
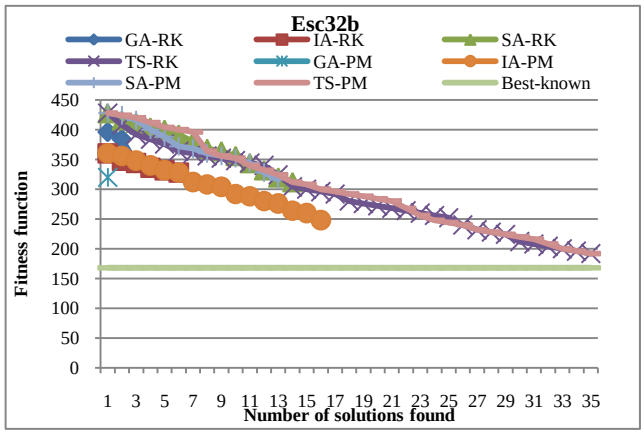


Figure 4.20: Search Trajectories of the Eight Chr* Files





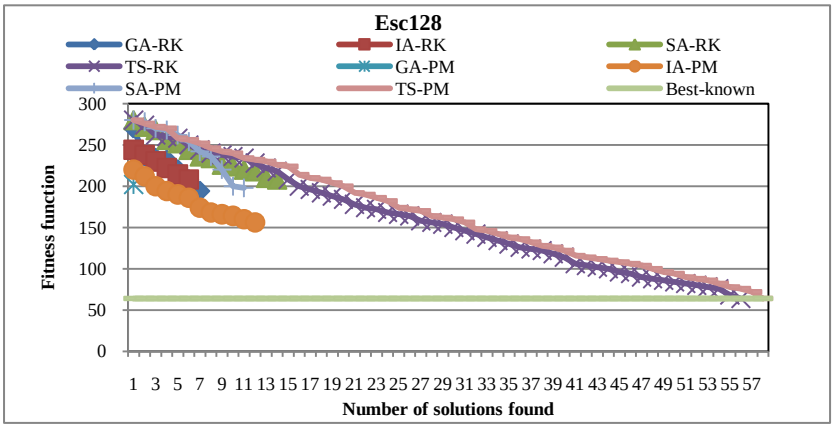
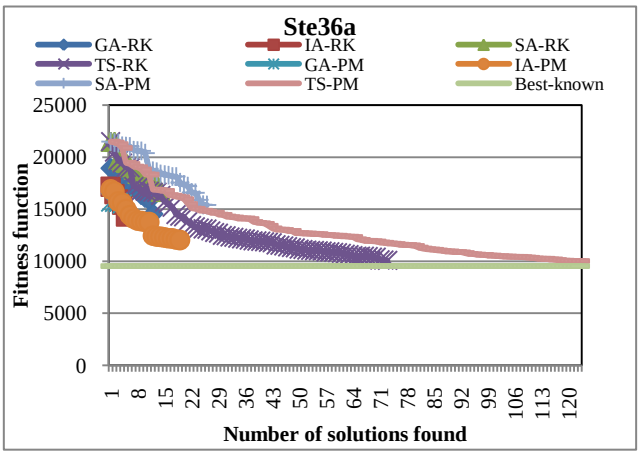
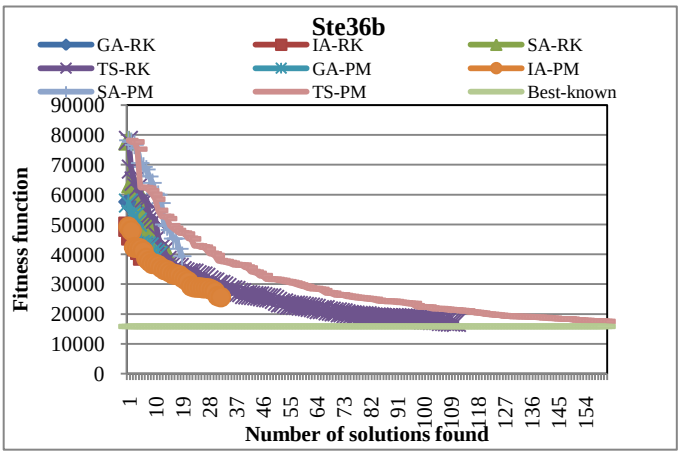


Figure 4.21: Search Trajectories of the Three Esc* Files





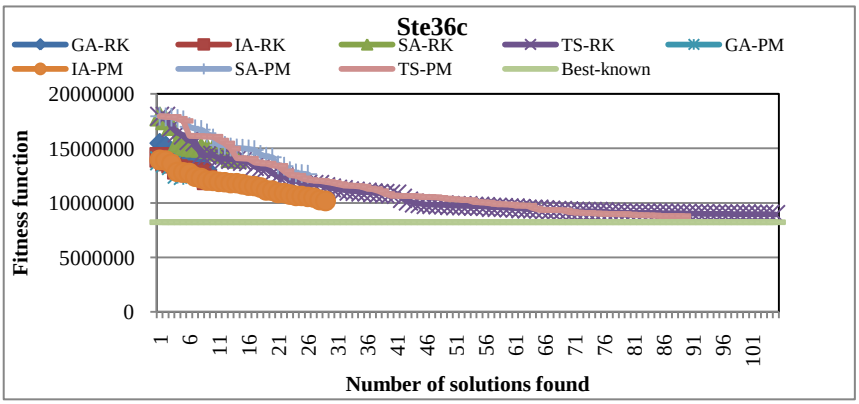


Figure 4.22: Search Trajectories of the Three Ste* Files

4.7 Conclusion

In chapter 4, the overall analysis clearly suggests that TS and IA-PM have a built-in capability to execute diversification and intensification mechanisms effectively unlike the other algorithms and that this feature appears to make TS and IA-PM very effective.

GAs and IAs tend to begin the search with more attractive solutions than SAs and TSs, mainly due to their population-based nature. However, as the number of fitness evaluations increases, TS takes over and the loss function drops more sharply than the rest. Furthermore, unlike TSs and IA-PM, the other algorithms show indications of stagnation in their early stages of execution. In TS, the stagnation effects are shown in later stages. In the case of IA-PM, stagnation is not quite visible in any part of the search trajectory. Similar observations are noted for all of the input files.

This study investigated how the performance of an algorithm is affected by problem characteristics using 14 hard problem instances for parameter tuning. The comparison studies addressed four research questions and led to the following findings.

First, we used the CART analysis tools to classify the problems in terms of the selected problem characteristics (size of the problem, flow and distance measures, sparsity of the matrices, and correlation of the flow and distance dominance measures). We used the 130 files as well as the 14 hard problems for this analysis. We identified sparsity as the prominent characteristic for determining performance. This phenomenon is true for the entire set of problems (130 files) as well as for the 14 hard problems. Higher sparsity measures indicated that the problems originated from

real-world data. Therefore, if a particular metaheuristic can sustain its performance with higher sparsity values, this indicates its appropriateness to solve similar real-world problems.

The findings of this chapter lay a foundation for further research studies on improving the individual search trajectories of less intelligent algorithms. More emphasis should be given to open new directions for the hybridizations of particular metaheuristics especially to solve real-world problems. We can further explore promising new research opportunities due to the power of grid-computing resources.

Utilizing efficient diversification and intensification methods, such as path-relinking to population-based algorithms, can generate better solutions. In the next chapter, such diversification and intensification mechanisms are explored within the context of QAPs.

CHAPTER FIVE

5. POSITION-BASED PATH RELINKING (POS_PR) AUGMENTATION

5.1 Introduction

In this chapter, we investigate the effects of Path-Relinking (PR) augmentation on the performance of each tuned path-based and population-based metaheuristic. Specifically, we look at the implementation of position-based PR (POS_PR) mechanisms on the eight metaheuristics discussed in chapters 3 and 4. This implementation process will be followed by a detailed analysis of how each method has improved and re-classify the input files according to trivial, moderate, and hard problems based on the new results.

The broad objectives of this chapter include the following:

- Implement of POS_PR for the two representation schemes
- Compare their performance against the generic-tuned implementations discussed in chapters 3 and 4.
- Classify problem instances of the QAPLIB based on classification and regression trees (CARTs)
- Compare the results with the problem instance classification discussed in chapter 3 and report any improvements

In this chapter we integrate the findings of chapters 3 and 4 in which we extensively tuned four metaheuristics using two representation schemes. In this chapter, we begin with the eight tuned metaheuristics and diversify and intensify the search trajectory using the aforementioned POS_PR mechanisms. The rest of the chapter unfolds as follows: in section 1, an introduction to the implementation of the PR mechanisms is given with respect to the two representation schemes. In section 2,

the results are compared, and in section 3, the classification tables are constructed. The chapter concludes with a chapter summary and conclusions that lay the foundation for chapter 6.

5.2 The Use of Diversification Mechanisms in Metaheuristics

Unlike exact mathematical models, the stochastic nature of metaheuristics requires efficient search mechanisms to explore the search space, and depending on the nature of the metaheuristic, the strategies utilized may differ. Specifically, the inspiration or the metaphor behind the implementation of a particular metaheuristic determines the extent of the diversification and intensification (D&I) of the search (Figure 5.1). Figure 5.1, illustrates that metaheuristic A has a stronger intensification capabilities to modify a given solution to find a better one in the neighborhood. However, it lacks the ability to explore diverse search regions at the same time. Therefore, it becomes trapped in one of the local optima.

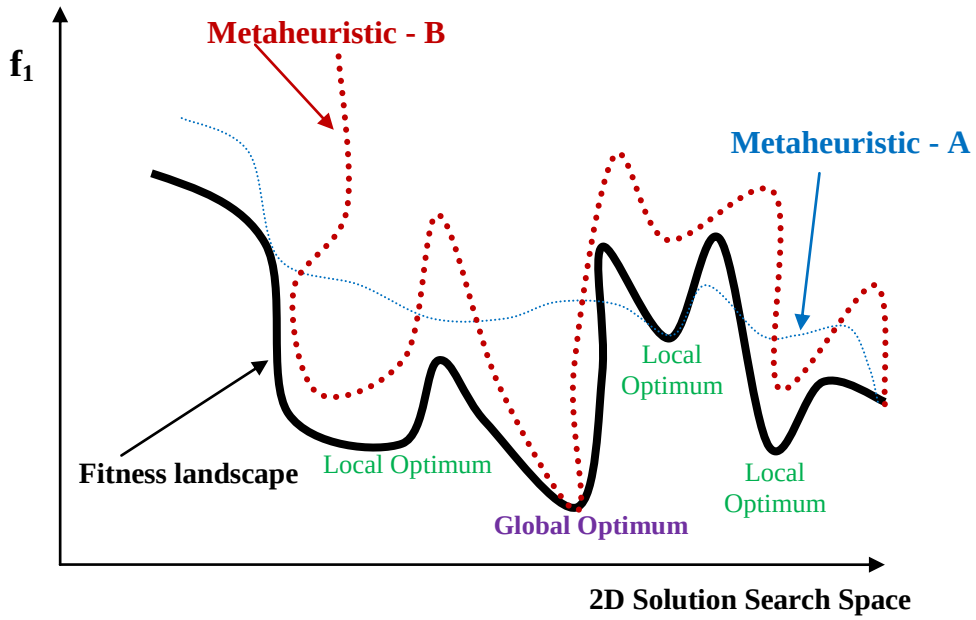


Figure 5.1: An Illustration of Diversification and Intensification Mechanisms

On the other hand, metaheuristic B—with its strong diversification mechanisms—explores different regions of the solution space and escapes from the local optima to find the best global solution. In other words, diversification is the capability of a particular metaheuristic to explore different search territories and intensification; on the other hand, however, is the ability to improve or modify a given solution to generate a better outcome [49].

From the previous chapter, it is evident that pure randomizations are not very effective at tackling hard problem instances. Therefore, investigating intelligent mechanisms that utilize strong intensification and diversification mechanisms is important and relevant. The objective of such investigations is to optimize the mapping process of the metaheuristic search to the actual shape of the fitness landscape for the problem at hand. PR is one such diversification mechanism that can be used to expand the search trajectory.

5.3 Implementation of POS_PR for Metaheuristics

The PR concept was first proposed by Glover [38] and stems from scatter search. PR embodies principles and strategies that are still not emulated by other evolutionary methods but are advantageous for solving a variety of complex problems. Like other evolutionary methods, PR operates with a population of solutions rather than with a single solution at a time and employs procedures for "combining" these solutions to create new ones [38]. One of the most distinguishing features of PR is its alliance with TS and its adoption of the principle that search can benefit by incorporating special forms of adaptive memory. For population-based metaheuristics, this can be incorporated via crossover and mutation operators, elite reproduction, combining generations, and populations of solutions through tunneling

and many other techniques. There are only a few variants of PR procedures proposed in the literature. A form of PR was proposed by [45] for QAP and recently, position-based PR and sequence-based PR were investigated by [93] for a multiple-facility layout problem. In the last section of this chapter, we compare our results with these PR-based variants to comment on the quality of solutions.

In this study, we have implemented POS_PR mechanisms for the two representation schemes. Figure 5.2 illustrates an implementation of the POS_PR mechanism. In this illustration, one solution is selected as the initial solution, and one of the elite solutions is selected as the guiding solution. The procedure begins by starting at the first position of the guiding solution and comparing it to the same position of the initial solution. If the two are the same, one can proceed to the next position. In this case, the guiding solution has number 6 (refer to Figure 5.2; first cell of the guiding solution) and the initial solution has number 1(refer again to Figure 5.2; first cell of the initial solution). Since the two numbers are not the same, we must look for number 6 in the initial solution, which is stored at position 6. Therefore, we swap numbers 6 and 1 to make the initial solution's first position the same as the guiding solution's first position. We repeat this procedure until we reach the last position of the guiding solution.

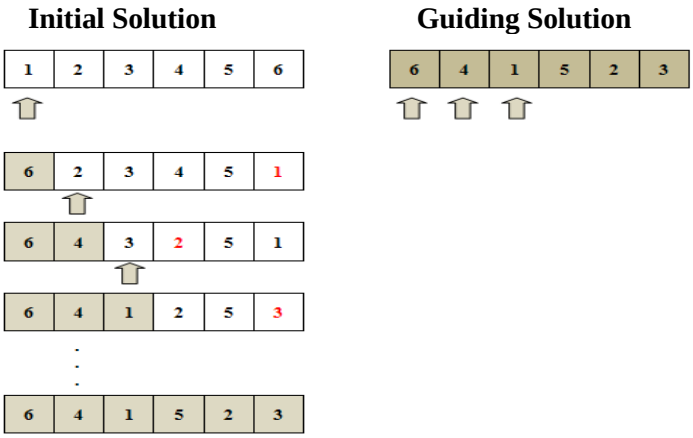


Figure 5.2: An Illustration of POS_PR Implementation

As we can see in Figure 5.2, the concept of POS_PR can easily be implemented for permutation-based representations. However, for random keys, such implementation requires some additional refinement as explained in section 5.2.1.

We consider two variants of POS_PR depending on the way the initial solution is selected, as depicted in Figure 5.3.

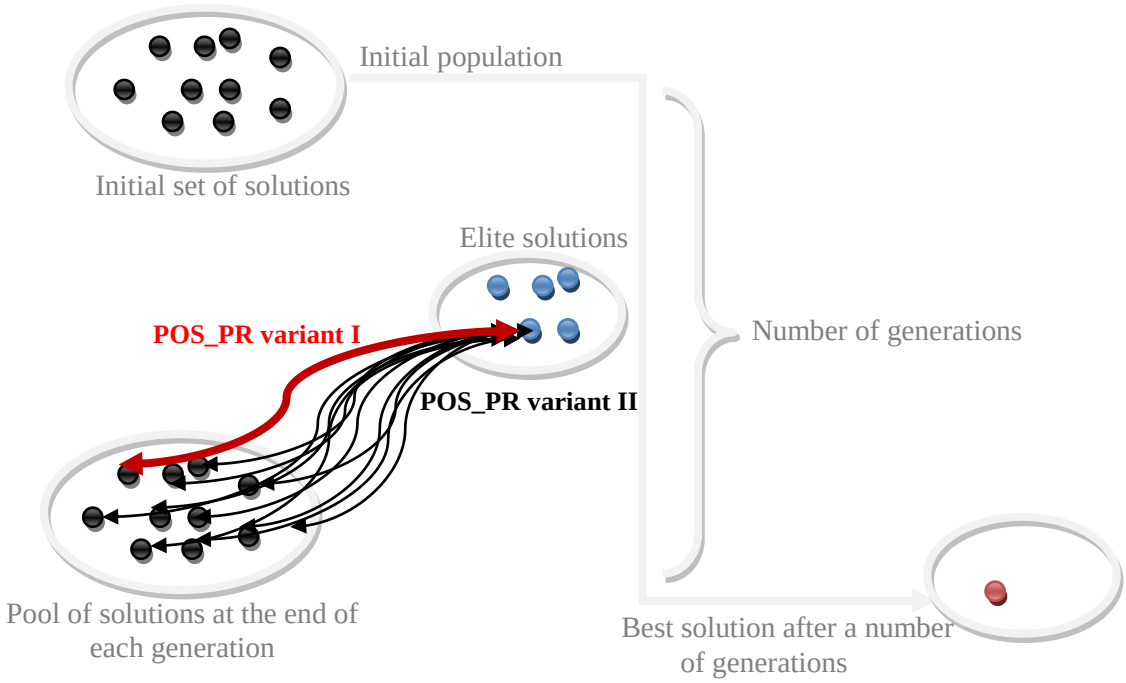


Figure 5.3: An Illustration of POS_PR for *Variant I* and *Variant II*

For *variant I*, out of the elite solutions, we randomly pick the guiding solution, and at every iteration, we path-relink the best solution for a particular generation's population. In *variant II*, the selection of guiding solutions remains the same; however, we path-relink every member of the population to diversify the search.

5.3.1 Implementation of POS_PR for Random Keys

Implementing POS_PR for random keys requires some additional considerations. In order to generate solutions in between the relinked path of the initial and guiding solutions, we use a position identifier. Figure 5.4 shows the pseudo code implementation of the random keys-based POS_PR.

Initialize the **population using random keys** (p_0) of candidate solutions $s_i \in S$

gen =0

While number of generations have not reached the maximum ($gen_j \leq gen_m$)

Find the best solution of $s_{best} \in S$

Find the generation best gen_{best}

If $s_{best} < gen_{\text{best}}$

Store this in the pool of guiding solutions ($best_{gen} \in G$)

For $s_i \in S$

Randomly select guiding elite solution from $best^*_{gen} \in G$

For each sequence of the guiding solution $x_k \in best^*_{gen}$

Compute identifiers for each sequence of $\boldsymbol{x_k^{ID} \epsilon\, best^*_{gen}}$

For each sequence of the $\boldsymbol{x_n \in s_i}$

Compare the sequences and the resulting random keys $\boldsymbol{best^*_{gen} \in G}$

Do until positions of identifiers of each sequence $\boldsymbol{s_i \equiv best^*_{gen}}$

Swap if $x_n \neq x_k$

Evaluate the intermediary solutions

Create a set of solutions (*m*) via crossover $S_m^{crossover}$ from $p_{gen\text{-}1}$

Create a set of solutions (*n*) from $S_m^{crossover}$ via mutation $S_n^{mutation}$

For each $s_i \in S, S_m^{crossover}, S_n^{mutation}$

Sort the Random Keys

Evaluate the fitness

$gen_{j+1} = gen_j + 1$

Figure 5.4: Pseudo Code of the Implementation of Random Keys-Based POS_PR *Variant II*

Figure 5.5 illustrate how random keys POS_PR implementation is implemented using an example.

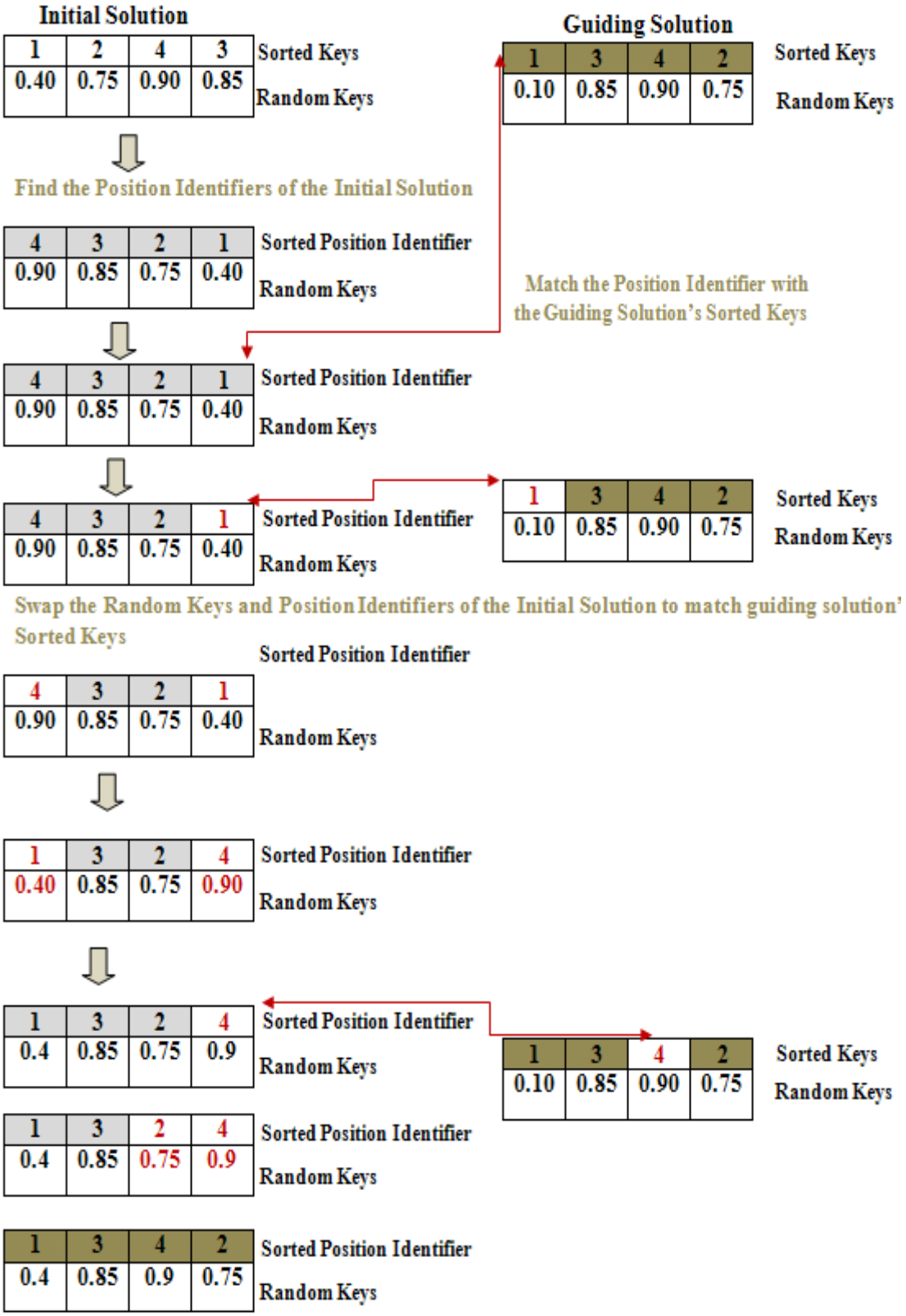


Figure 5.5: An Illustration of POS_PR Implementation

5.3.2 Implementation of POS_PR for Permutation

The POS_PR for permutation-based metaheuristics followed the same procedure explained in Figure 5.2. Since the solution representation itself dealt with numbers that are interchangeable, it did not cause any additional constraints. Figure 5.6 shows the pseudo code implementation of the permutation-based POS_PR.

Initialize the population using **permutation** (p_0) of candidate solutions $s_i \in S$

gen =0

While number of generations have not reached the maximum ($gen_j \leq gen_m$)

Find the best solution of $s_{best} \in S$

Find the generation best gen_{best}

If $s_{best} < gen_{best}$

Store this in the pool of guiding solutions ($best_{gen} \in G$)

For $s_i \in S$

Randomly select guiding elite solution from $best^*_{gen} \in G$

For each sequence of the guiding solution $x_k \in best^*_{gen}$

For Variant I **For each sequence of the** $\boldsymbol{x_n^{BEST}} \in \boldsymbol{s_i}$

For Variant II **For each sequence of the** $\boldsymbol{x_n} \in \boldsymbol{s_i}$

Compare the sequences and the resulting solution $\boldsymbol{best^*_{gen}} \in \boldsymbol{G}$

Swap if $x_n \neq x_k$ **or** $\boldsymbol{x_n^{BEST}} \neq x_k$

Evaluate the intermediary solutions

Create a set of solutions (*m*) via crossover $S_m^{crossover}$ from $p_{gen\text{-}1}$

Create a set of solutions (*n*) from $S_m^{crossover}$ via mutation $S_n^{mutation}$

For each $s_i \in S, S_m^{crossover}$,

Evaluate the fitness

$gen_{j+1} = gen_j + 1$

Figure 5.6: Pseudo Code of the Implementation of Permutation-Based POS_PR *Variant I* and *Variant II*

In the following sections, the results of the comparison studies will be presented. The performance of the four tuned random keys-based metaheuristics and the four tuned permutation-based metaheuristics will be compared with their POS_PR-augmented implementations. The performance of each metaheuristic will be

evaluated based on problem characteristics, and the results will be compared with the input-file classifications presented in chapters 3 and 4. Any improvements in the solution quality or performance will be presented separately for path-based metaheuristics and population-based metaheuristics. Section 5.3 presents the results of the comparison study for random keys.

5.4 Comparison: Random Keys

In the comparison study for the random keys-based metaheuristics, the performance of the tuned population-based methods (GA-RK and IA-RK) will be compared against their POS_PR augmentations. We use the moderate and the hard problems identified in chapters 3 and 4 for this analysis. In the following section, the two path-based methods (TS-RK and SA-RK) will be analyzed using the same process of the aforementioned test cases. All the graphs presented here depict average loss functions values over 50 replications.

5.4.1 Path-Based Methods

The path-based methods showed some erratic behavior as depicted in graphs 5.7 and 5.8. From Figure 5.8, it is evident that POS_PR augmentation did not improve the performance of either SA or TS.

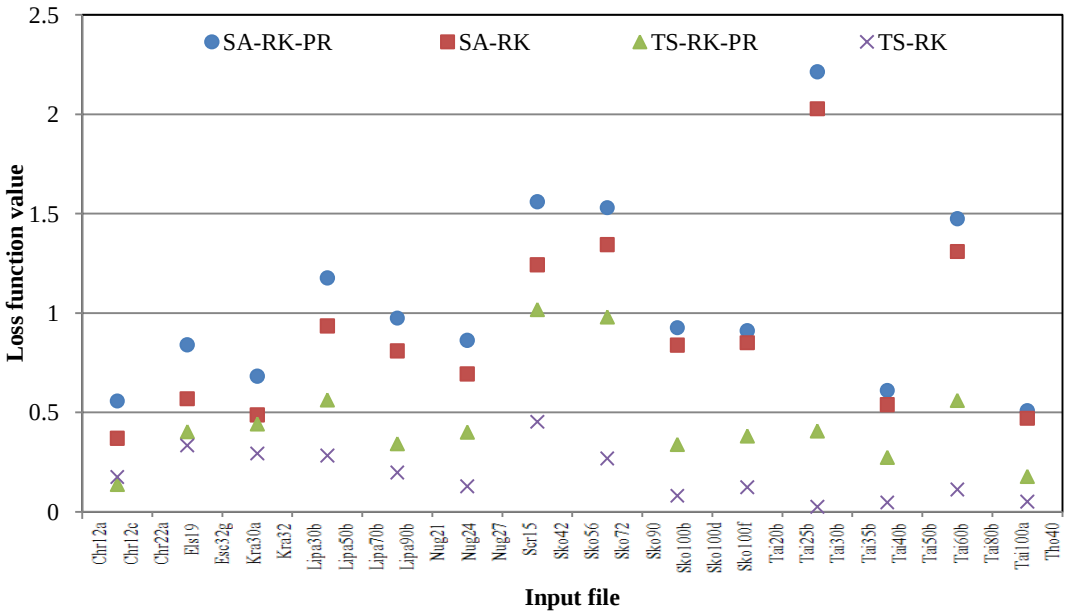


Figure 5.7: The Performance of POS_PR_SA-RK, POS_PR_TS-RK, SA-RK, and TS-RK for the 63 Moderate Problems

The same analysis was carried out for the 14 hard problems, the results of which are depicted in Figure 5.8. The same scenario can be seen for SA and TS. Based on the results, PR–based diversification mechanisms are not suited for random keys-based SAs or TSs implemented in the QAP context. The performance of the SAs and TSs were not significantly improved by the POS_PR diversification/intensification mechanisms but were actually adversely affected by these mechanisms as Figures 5.7 and 5.8 show.

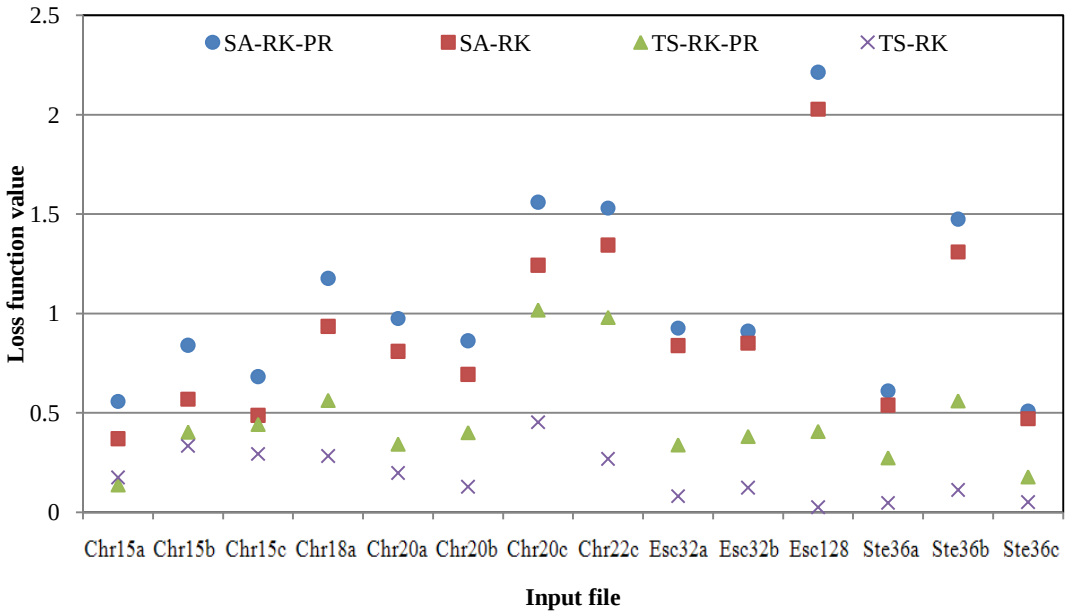


Figure 5.8: The Performance of POS_PR_SA-RK, POS_PR_TS-RK, SA-RK, and TS-RK for the 14 Hard Problems

5.4.2 Population-Based Methods

The performance of POS_PR_GA-RK and POS_PR_IA-RK will be compared with GA-RK and IA-RK. Figure 5.9 depicts the performance of these four methods for the 63 moderate problems discussed in chapter 3.

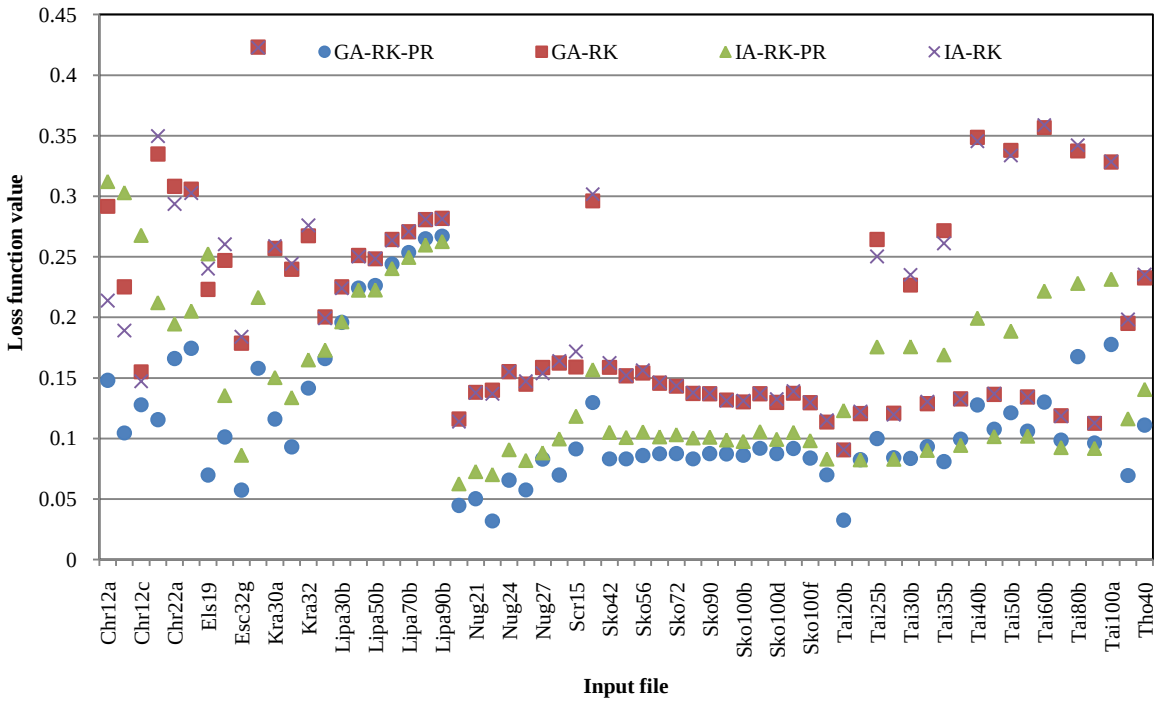


Figure 5.9: The Performance of POS_PR_GA-RK, POS_PR_IA-RK, GA-RK, and IA-RK for the 63 Moderate Problems

It is clearly visible from Figure 5.9 that GA-RK and IA-RK drastically improved with the POS_PR augmentation. Many of the problems that were identified as moderate problems in the chapter 3 can be re-classified as trivial problems due to this performance improvement. The GAs showed significantly more improvement than the IAs; thus, POS_PR can be considered a well-suited diversification mechanism to expand the search space.

The same analysis was carried out for the 14 hard problems. Figure 5.10 depicts the performance of the two population-based random keys implementations.

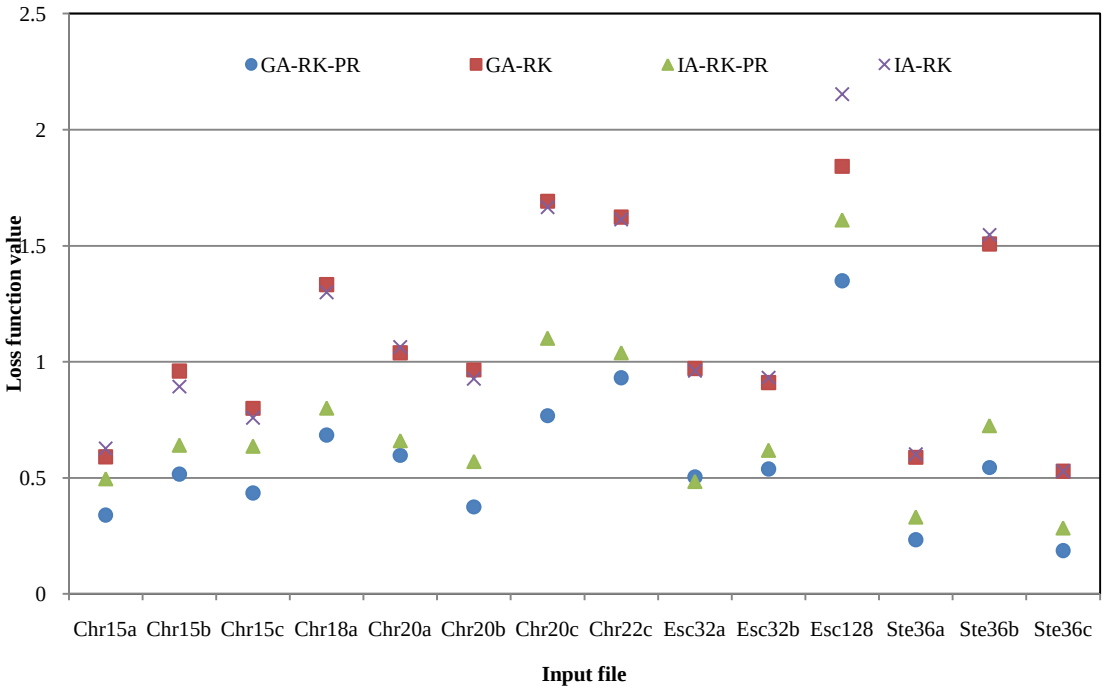


Figure 5.10: The Performance of POS_PR_GA-RK, POS_PR_IA-RK, GA-RK, and IA-RK for the 14 Hard Problems

POS_PR significantly improved the performance of the population-based metaheuristics for the hard problem instances with the improvement varying between 25.16% and 69.25%. The performance improvements for the GAs were slightly higher than that for the IAs, as observed in the moderate problem analysis as well.

In order to present a summary of the results obtained by these two analyses, we created Table 5.1. Throughout the 63 problem instances, the GAs showed positive improvement. Considering the problem categories, the GAs' performance was significantly improved for the Chr*, Els*, Esc* Nug*, Sko*, Tai*, and Tho* problems. In the case of IA, even though POS_PR helped this method to perform better in most of the problem categories, for Chr* it caused a slight deterioration in performance. Considering its overall performance, POS_PR can be considered a promising method for improving random keys population-based metaheuristics. In the case of path-based metaheuristics, as Table 5.1 shows, the SAs' and TSs' performance were adversely affected by the augmentation. Except for a few problem

instances, most of these two groups' performance was negatively impacted. Hence, POS_PR is not very suitable for random keys-based SA/TS in the QAP context. The performance improvement was computed using the following formula.

$$GA\_RK\ loss - GA\_RK\_PR\ loss \quad (1)$$

$$IA\_RK\ loss - IA\_RK\_PR\ loss \quad (2)$$

$$SA\_RK\ loss - SA\_RK\_PR\ loss \quad (3)$$

$$TS\_RK\ loss - TS\_RK\_PR\ loss \quad (4)$$

Table 5.1 Summary of the Performance Improvements of the 63 Moderate Problems Using POS_PR for Random Keys Representation

Problem Name	Difference in performance improvement (+) / deterioration (-) as a percentage			
	GA	IA	SA	TS
Chr12a	14.36%	-9.83%	-8.84%	-1.18%
Chr12b	12.06%	-11.38%	-9.44%	27.57%
Chr12c	2.71%	-12.06%	-7.29%	1.22%
Chr18b	21.94%	13.75%	-8.99%	-2.10%
Chr22a	14.23%	9.90%	-4.97%	-4.22%
Chr22b	13.13%	9.73%	-3.73%	-6.82%
Els19	15.33%	-1.22%	-7.09%	9.61%
Esc32d	14.58%	12.48%	-4.00%	0.42%
Esc32g	12.14%	9.77%	-3.17%	2.30%
Esc64a	26.52%	20.62%	-8.00%	0.00%
Kra30a	14.08%	10.85%	-2.57%	-3.97%
Kra30b	14.67%	11.03%	-1.79%	-8.18%
Kra32	12.58%	11.09%	-2.23%	-9.85%
Lipa20b	3.43%	2.63%	-0.99%	-9.73%
Lipa30b	2.93%	2.76%	-0.67%	-6.28%
Lipa40b	2.69%	2.76%	-0.49%	-6.22%
Lipa50b	2.19%	2.58%	-0.41%	-4.55%
Lipa60b	2.04%	2.31%	-0.27%	16.70%
Lipa70b	1.70%	2.12%	-0.27%	18.20%
Lipa80b	1.57%	2.11%	-0.24%	19.19%
Lipa90b	1.46%	1.86%	-0.20%	9.64%
Nug20	7.14%	5.12%	-1.64%	-0.78%
Nug21	8.79%	6.50%	-2.36%	-3.92%
Nug22	10.79%	6.66%	-1.97%	-1.07%
Nug24	8.96%	6.41%	-1.82%	-3.14%
Nug25	8.73%	6.52%	-1.82%	-4.05%
Nug27	7.56%	6.56%	-1.79%	-4.41%
Nug30	9.25%	6.43%	-1.19%	-7.26%
Scr15	6.77%	5.34%	-4.15%	-3.15%
Scr20	16.64%	14.49%	-4.15%	-12.06%
Sko42	7.56%	5.71%	-1.05%	-5.70%
Sko49	6.85%	5.11%	-0.86%	-5.88%
Sko56	6.82%	5.07%	-0.90%	-6.55%
Sko64	5.84%	4.47%	-0.62%	-5.97%
Sko72	5.57%	4.01%	-0.47%	-6.22%
Sko81	5.40%	3.73%	-0.47%	-4.24%
Sko90	4.94%	3.59%	-0.38%	-6.30%
Sko100a	4.45%	3.24%	-0.41%	-6.07%
Sko100b	4.41%	3.35%	-0.35%	-5.93%
Sko100c	4.51%	3.13%	-0.45%	-7.47%
Sko100d	4.24%	3.31%	-0.42%	-5.72%
Sko100e	4.56%	3.39%	-0.43%	-6.90%
Sko100f	4.57%	3.15%	-1.38%	-6.25%

Tai20a	4.36%	3.18%	-1.20%	-1.93%
Tai20b	5.80%	-3.23%	-1.68%	4.94%
Tai25a	3.80%	3.97%	-1.35%	-5.59%
Tai25b	16.43%	7.46%	-6.12%	1.97%
Tai30a	3.67%	3.67%	-1.00%	-3.41%
Tai30b	14.32%	5.92%	-6.23%	-0.81%
Tai35a	3.54%	4.00%	-0.95%	-4.55%
Tai35b	19.06%	9.20%	-3.99%	-1.18%
Tai40a	3.34%	3.80%	-0.98%	-5.39%
Tai40b	22.11%	14.62%	-3.17%	-3.66%
Tai50a	2.89%	3.53%	-0.82%	-5.78%
Tai50b	21.68%	14.50%	-3.53%	-4.57%
Tai60a	2.83%	3.20%	-0.64%	-6.19%
Tai60b	22.65%	13.71%	-1.17%	5.32%
Tai80a	2.03%	2.59%	-0.78%	-6.29%
Tai80b	16.98%	11.40%	-0.97%	-5.58%
Tai100b	1.62%	2.09%	-0.90%	-4.92%
Tai100a	15.06%	9.71%	-1.23%	-6.90%
Tho30	12.56%	8.19%	-2.11%	--7.12%
Tho40	12.15%	9.48%	-1.34%	-2.75%

The Table 5.2 depicts the summary of the comparison for the 14 hard problems. As noted in the previous table, the GAs shows a positive significant improvement throughout the three problem categories. Unlike the moderate problems, the POS_PR-augmented IA metaheuristics demonstrate the ability to perform well. However, the SAs and TSs present a totally different scenario, yielding deteriorations for all of the problem instances.

Table 5.2 Summary of the Performance Improvements of the 14 Hard Problems Using POS_PR for Random Keys Representation

Problem Name	Difference in performance improvement (+) / deterioration (-) as a percentage			
	GA	IA	SA	TS
Chr15a	25.16%	13.00%	-18.74%	3.81%
Chr15b	44.41%	25.25%	-27.17%	-6.91%
Chr15c	36.42%	12.25%	-19.48%	-14.84%
Chr18a	64.82%	49.84%	-24.19%	-27.98%
Chr20a	44.17%	40.36%	-16.50%	-14.47%
Chr20b	59.03%	35.61%	-16.89%	-27.19%
Chr20c	92.44%	56.40%	-31.75%	-56.47%
Chr22c	69.25%	57.40%	-18.61%	-71.26%
Esc32a	46.74%	47.60%	-8.80%	-25.75%
Esc32b	37.29%	31.29%	-6.05%	-25.71%
Esc128	49.25%	54.19%	-18.56%	-38.13%
Ste36a	35.54%	27.06%	-7.15%	-22.61%
Ste36b	96.33%	82.17%	-16.54%	-44.80%
Ste36c	34.16%	24.42%	-3.75%	-12.63%

5.5 Comparison: Permutations

In the comparison study for the permutation-based metaheuristics, the performances of the tuned population-based methods (GA-PM and IA-PM) are compared with their POS_PR augmentations. For this analysis, we used the moderate and hard problems identified before and conducted a comparison study similar to the one presented in the previous sections. This analysis is followed by the section on the two path-based methods (TS-PM and SA-PM).

5.5.1 Path-Based Methods

Similar to the random keys POS_PR implementation, SA shows marginal improvement (see Figure 5.11). It is evident from these results and from the previous findings that POS_PR augmentation is not suitable for SA irrespective of the representation scheme. However, TS shows some significant improvement overall and some marginal improvement for a few instances, especially for the Sko*, and Tai* problem categories.

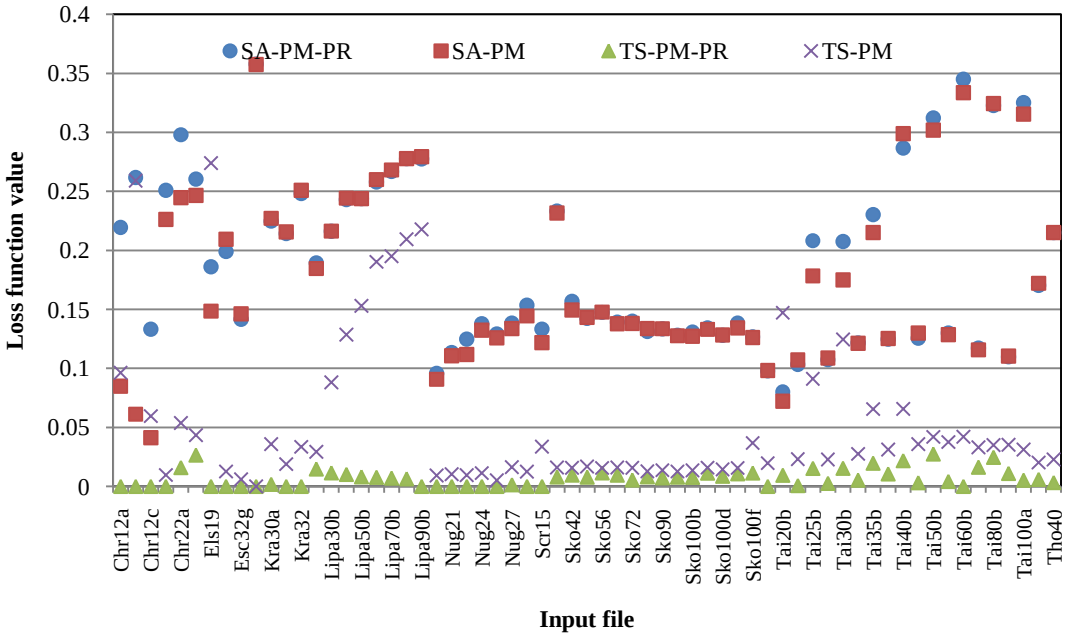


Figure 5.11: The Performance of POS_PR_SA-PM, POS_PR_TS-PM, SA-PM, and TS-PM for the 63 Moderate Problems

As shown in Figure 5.12, it is evident that the solution quality of SA was not improved by the POS_PR augmentation as was the case in the previous cases. However, for TS, the POS_PR augmentation certainly improves the solutions' quality. It is more appropriate to conclude that the new diversification mechanisms, including POS_PR, enhance the exploratory abilities of TS with respect to permutation-based implementations as opposed to random keys.

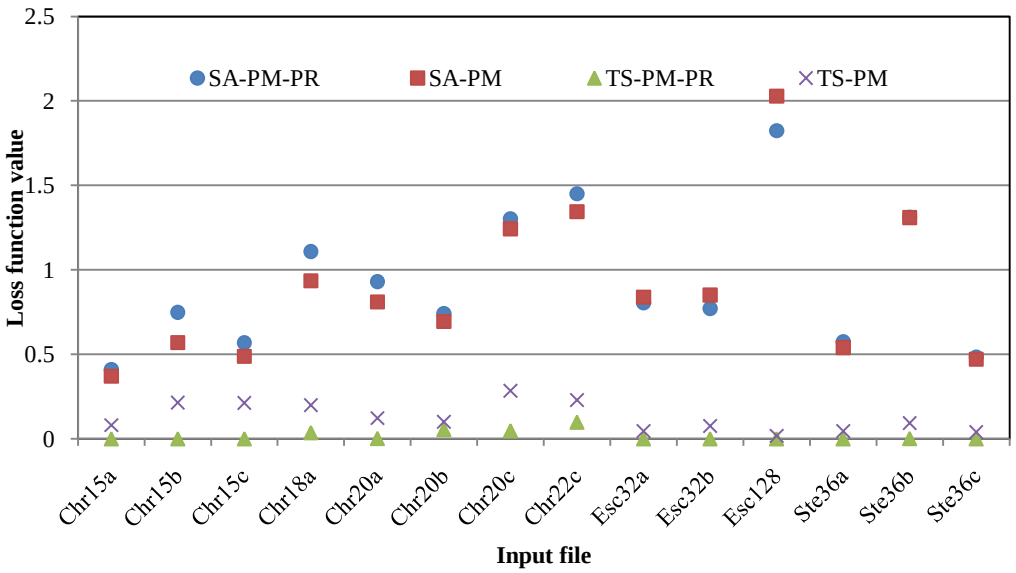


Figure 5.12: The Performance of POS_PR_SA-PM, POS_PR_TS-PM, SA-PM, and TS-PM for the 14 Moderate Problems

5.5.2 Population-Based Methods

Figure 5.13 shows that GA-PM was significantly improved for the 63 moderate problem instances except for the seven Lipa* problems. These problems were pseudo-randomly generated with known optimal solutions [18]. It is surprising to see that the performance of all of the other methods, including IA-PM and the POS_PR augmentations of the GA and IA, deteriorated for these problems. Unlike the random keys implementation, IA did not gain any significant improvements from the new diversification mechanism.

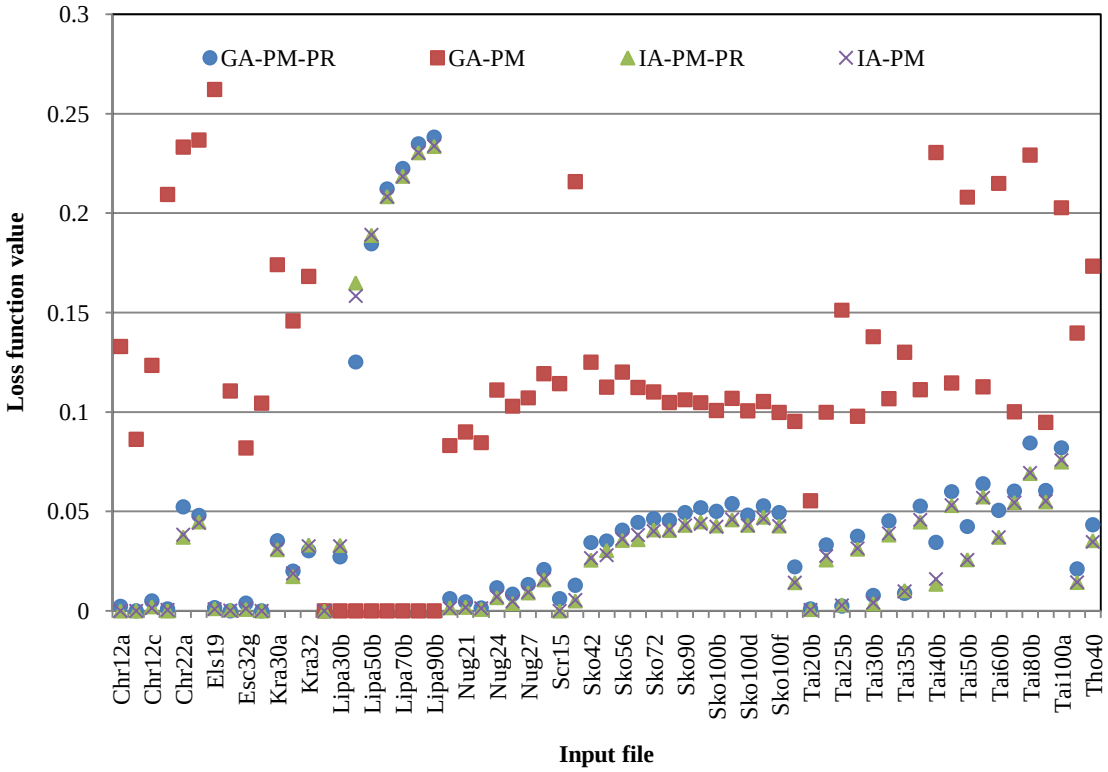


Figure 5.13: The Performance of POS_PR_GA-PM, POS_PR_IA- PM, GA- PM, and IA- PM for the 63 Moderate Problems

The same analysis has been carried out with the 14 hard problems and the results are depicted in Figure 5.14. Similar to the previous case, POS_PR augmentation to GA has outperformed the GA-PM. However, IA-PM was as good as its POS_PR augmentation.

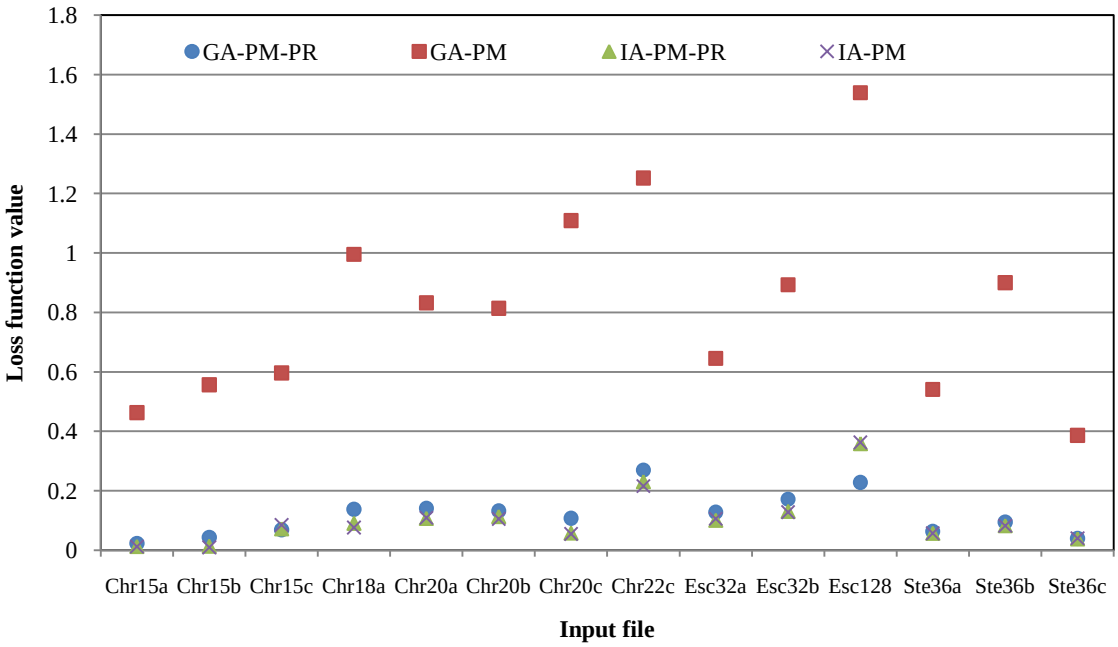


Figure 5.14: The Performance of POS_PR_GA-PM, POS_PR_IA- PM, GA- PM, and IA- PM for 14 Hard Problems

It can be seen that even though POS_PR significantly improved both GA-RK and IA-RK, the permutation-based IA was unaffected by the same procedure. As such, we can conclude that once a particular method has fully exploited the ability to diversify by nature or by the implemented representation scheme, an attempt to embed additional diversification/intensification creates marginal returns. Tables 5.3 and 5.4 summarize the results of these analyses and performance improvement is calculated using following formula.

$$GA\_PM\ \ loss - GA\_PM\ \_PR\ loss \quad (5)$$

$$IA\_PM\ \ loss - IA\_PM\ \_PR\ loss \quad (6)$$

$$SA\_PM\ \ loss - SA\_PM\ \_PR\ loss \quad (7)$$

$$TS\_PM\ \ loss - TS\_PM\ \_PR\ loss \quad (8)$$

Table 5.3 Summary of the Performance Improvement of Moderate Problems Using POS_PR for Permutation Representation

Problem Name	Difference in performance improvement (+) / deterioration (-) as a percentage			
	GA	IA	SA	TS
Chr12a	13.06%	0.00%	-13.45%	9.63%
Chr12b	8.62%	0.00%	-20.07%	25.90%
Chr12c	11.85%	-0.07%	-9.19%	5.95%
Chr18b	20.83%	0.05%	-2.45%	0.96%
Chr22a	18.10%	0.12%	-5.34%	3.79%
Chr22b	18.87%	-0.06%	-1.38%	1.69%
Els19	26.05%	-0.03%	-3.76%	27.40%
Esc32d	11.06%	-0.04%	1.04%	1.26%
Esc32g	7.81%	0.03%	0.47%	0.59%
Esc64a	10.45%	0.00%	-0.03%	0.00%
Kra30a	13.88%	0.04%	0.24%	3.41%
Kra30b	12.57%	0.15%	0.15%	1.89%
Kra32	13.80%	-0.06%	0.27%	3.35%
Lipa20b	0.00%	0.00%	-0.47%	1.45%
Lipa30b	-2.71%	-0.05%	0.01%	7.68%
Lipa40b	-12.51%	-0.67%	0.11%	11.85%
Lipa50b	-18.46%	0.02%	0.00%	14.49%
Lipa60b	-21.22%	-0.01%	0.19%	18.26%
Lipa70b	-22.24%	-0.02%	0.14%	18.81%
Lipa80b	-23.49%	-0.01%	0.02%	20.30%
Lipa90b	-23.83%	0.02%	0.19%	21.78%
Nug20	7.69%	-0.04%	-0.51%	0.92%
Nug21	8.54%	-0.01%	-0.28%	1.04%
Nug22	8.32%	0.04%	-1.30%	0.96%
Nug24	9.94%	0.03%	-0.54%	1.13%
Nug25	9.45%	0.11%	-0.34%	0.52%
Nug27	9.38%	0.03%	-0.46%	1.49%
Nug30	9.85%	0.02%	-0.90%	1.25%
Scr15	10.81%	0.00%	-1.15%	3.37%
Scr20	20.30%	0.04%	-0.17%	0.81%
Sko42	9.07%	0.09%	-0.74%	0.62%
Sko49	7.74%	-0.24%	0.09%	0.90%
Sko56	7.95%	0.07%	0.03%	0.42%
Sko64	6.78%	0.22%	-0.13%	0.66%
Sko72	6.37%	-0.07%	-0.19%	1.02%
Sko81	5.93%	0.02%	0.26%	0.47%
Sko90	5.67%	-0.02%	0.01%	0.65%
Sko100a	5.28%	-0.11%	-0.03%	0.48%
Sko100b	5.07%	-0.05%	-0.37%	0.58%
Sko100c	5.30%	0.01%	-0.12%	0.47%
Sko100d	5.24%	-0.01%	0.02%	0.57%
Sko100e	5.25%	-0.07%	-0.42%	0.50%
Sko100f	5.03%	-0.02%	-0.06%	2.54%
Tai20a	7.31%	0.00%	0.05%	1.97%
Tai20b	5.43%	-0.02%	-0.79%	13.78%
Tai25a	6.66%	0.19%	0.38%	2.24%
Tai25b	14.88%	-0.04%	-2.98%	7.60%
Tai30a	6.03%	0.03%	0.16%	2.03%
Tai30b	13.00%	0.04%	-3.25%	10.93%
Tai35a	6.15%	0.07%	-0.05%	2.23%
Tai35b	12.12%	-0.05%	-1.53%	4.60%
Tai40a	5.85%	0.10%	0.06%	2.06%
Tai40b	19.60%	0.25%	1.21%	4.40%
Tai50a	5.46%	0.01%	0.43%	3.28%
Tai50b	16.56%	-0.01%	-1.03%	1.46%
Tai60a	4.87%	-0.05%	-0.14%	3.34%
Tai60b	16.44%	-0.01%	-1.14%	4.20%
Tai80a	3.98%	-0.01%	-0.15%	1.70%
Tai80b	14.47%	0.03%	0.17%	1.04%
Tai100b	3.43%	0.04%	0.07%	2.43%
Tai100a	12.07%	0.10%	-0.98%	2.60%
Tho30	11.85%	0.00%	0.18%	1.43%
Tho40	13.01%	-0.07%	0.01%	1.96%

Table 5.4 Summary of the Performance Improvement of Hard Problems Using POS_PR for Permutation Representation

Problem Name	Difference in performance improvement (+) / deterioration (-) as a percentage			
	GA-PM	IA- PM	SA- PM	TS- PM
Chr15a	43.98%	-0.01%	-3.94%	8.02%
Chr15b	51.33%	-0.42%	-17.85%	21.40%
Chr15c	52.76%	1.27%	-8.05%	21.24%
Chr18a	85.74%	-1.43%	-17.30%	16.32%
Chr20a	69.12%	0.20%	-12.04%	11.98%
Chr20b	68.14%	-0.75%	-4.64%	4.59%
Chr20c	100.10%	-0.29%	-5.79%	23.69%
Chr22c	98.25%	-1.48%	-10.61%	13.11%
Esc32a	51.75%	0.52%	3.35%	4.52%
Esc32b	72.10%	-0.29%	7.95%	7.48%
Esc128	131.06%	0.44%	20.44%	1.63%
Ste36a	47.78%	-0.01%	-3.49%	4.46%
Ste36b	80.46%	-0.04%	-0.25%	9.13%
Ste36c	34.64%	0.21%	-1.32%	3.87%

The comparison studies carried out for the two representation schemes showed that in order for the POS_PR implementation to be successful, the implementer should carefully evaluate the philosophical differences (i.e., path-based metaheuristic or population-based metaheuristic) and the representation scheme that were utilized. It was evident that from the 77 problem instances considered, SA yielded marginal returns irrespective of the representation scheme. In the path-based analysis, it was found out that the POS_PR implementations are more beneficial when permutation representations are used rather than random keys. The random keys GA and IA were significantly improved by POS_PR, and the POS_PR-augmented TS-PM outperformed the rest of the augmentations with improvement in the solution quality. This metaheuristic was able to generate <0.5% of the best known value for almost all of the instances out of the 63 moderate problems and <1% of the best known values for the 14 hard problems.

5.6 Classification using CARTs

In this section, we investigate the criterion taken into consideration when the quality of a particular solution is determined. Figures 5.15-5.16 show the rankings of each criterion for representation based on the philosophical differences for the moderate and hard problems discussed previously.

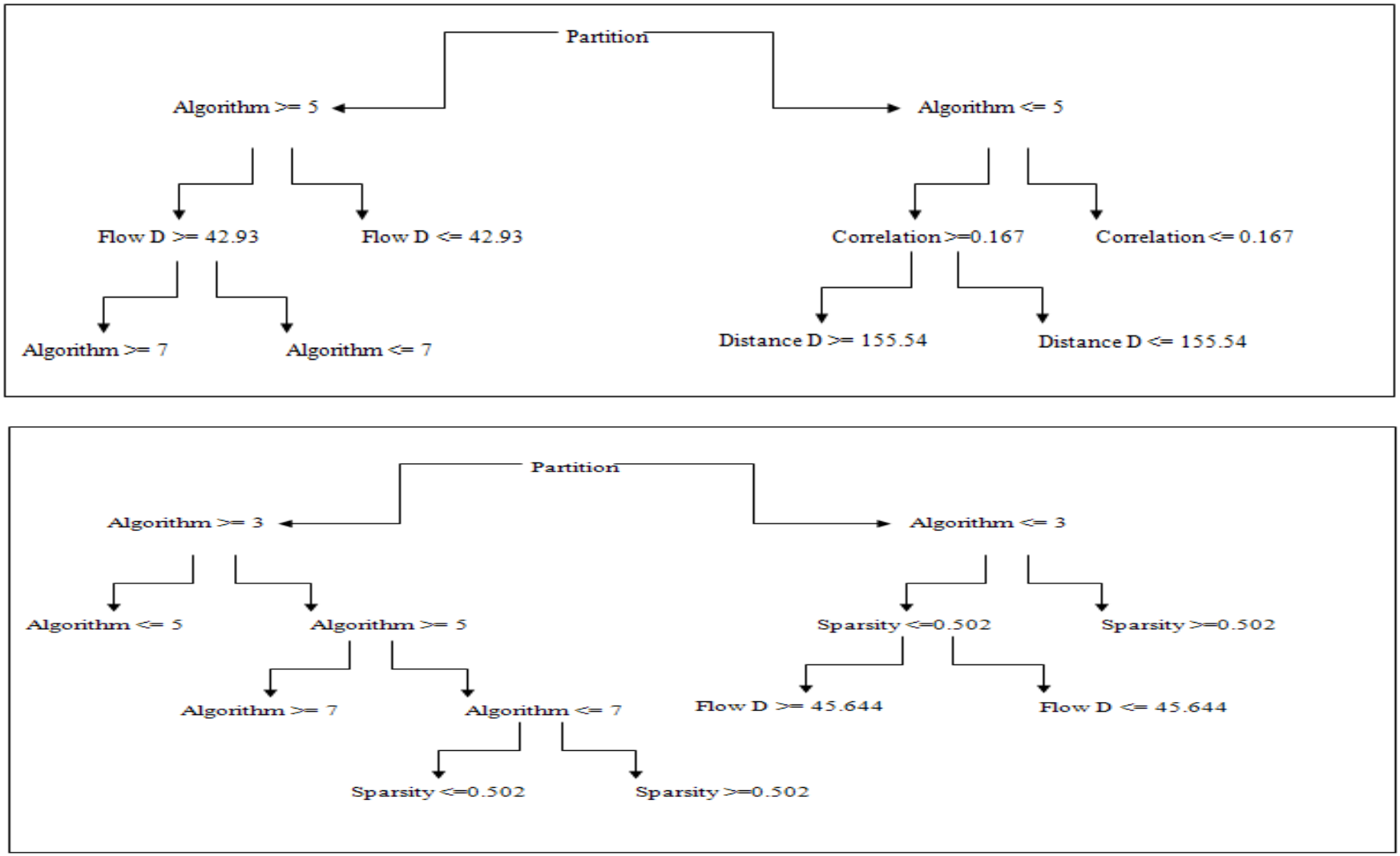


Figure 5.15: Partition by Metaheuristic Method for the 63 Moderate Problems
Top: Population-Based Methods (GARK-PR-1, GARK-2, IARK-PR-3, IARK-4, GAPM-PR-5, GAPM-6, IAPM-PR-7, and IAPM -8**)**
Bottom: Path-Based Methods (SARK-PR-1, SARK-2, TSPM-PR-3, TSPM-4, SAPM-PR-5, SAPM-6, TSRK-PR-7, and TSRK -8**)**

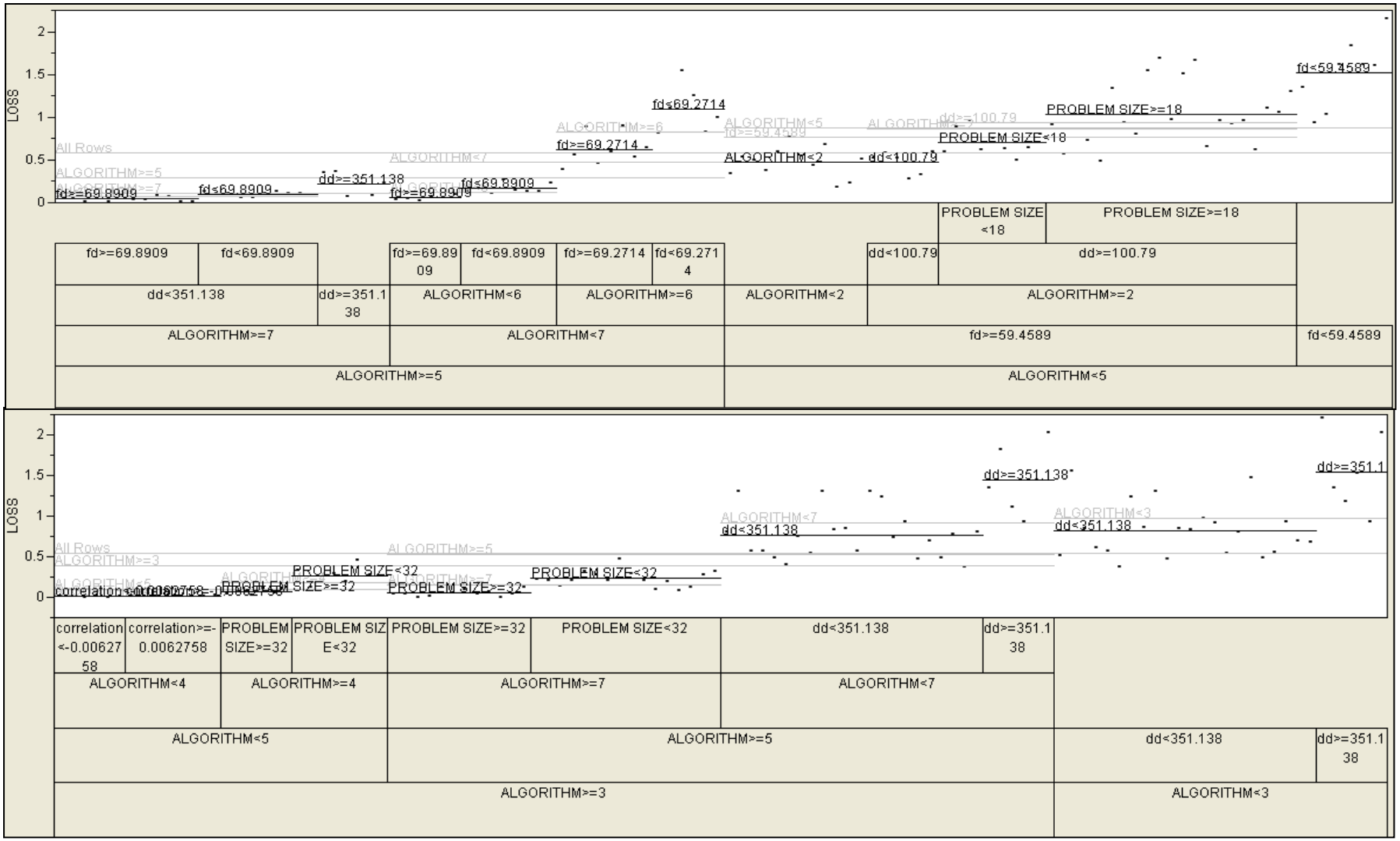


Figure 5.16: Partition by Metaheuristic Method for 14 Hard Problems.
Top: Population-Based Methods (GARK-PR-1, GARK-2, IARK-PR-3, IARK-4, GAPM-PR-5, GAPM-6, IAPM-PR-7, and IAPM -8**)**
Bottom: Path-Based Methods (SARK-PR-1, SARK-2, TSPM-PR-3, TSRK-4, SAPM-PR-5, SAPM-6, TSRK-PR-7, and TSPM -8**)**

From Figures 5.15 and 5.16, it is evident that the underlining criterion for determining solution quality varies drastically depending on the metaheuristic method. In Figure 5.15, the population-based metaheuristics are compared with the two representation schemes for the 63 moderate problems. The prominent criterion for partition is the representation method; the CART analysis shows "Algorithm >=5," and according to the legend, key 5 and higher are the permutation-based implementations. Another interesting finding is that "Algorithm>=7" is also considered as a partitioning criterion. This means that methods that are represented by legend keys 7 and 8 yield better solutions than the rest of the methods (Algorithm>=7: IAPM-PR-7 and IAPM-8). Figure 5.15 also shows that in the path-based analysis, the first splitting criterion for partitioning was "Algorithm>=3," referring to SARK-PR, SARK, and TSPM-PR (SARK-PR-1, SARK-2, TSPM-PR-3). These methods yielded the best quality solutions.

When Figure 5.16 is considered, the population-based methods determined that the most prominent splitting criterion was "Algorithm>=5." This means that there exists a difference in solution quality between the following methods: GAPM-PR, GAPM, IAPM-PR, IAPM, and the rest. In the lower levels of the splitting criteria hierarchy, the CART analysis yielded "Algorithm>=7," which can be interpreted as IAPM-PR making IAPM the most competitive method in the analysis.

For the path-based metaheuristic analysis, the first splitting criterion can be identified as "Algorithm>=3," referring to SARK-PR, SARK, and TSPM-PR (SARK-PR-1, SARK-2, TSPM-PR-3). It can be observed from Figures 5.15 and 5.16 that the best performing methods are similar for the hard and moderate problems. However, when the problem characteristics are considered, for the population-based methods,

the sparsity of the data matrices determines whether a particular problem is a moderate or a hard problem to be solved. For path-based metaheuristics, distance dominance and the size of the problem play a major role as opposed to the sparisity of the matrices.

In the following section, we compare the best POS_PR-augmented methods indentified in this chapter with a few state-of-the-art algorithms previously developed for QAPs.

5.7 Comparison with state-of-the-art PR-based methods

In the literature, there are few prominent studies that have investigated the QAPs using PR-based diversification strategies. Out of these studies, the Greedy Genetic Algorithms developed by Ahuja *et al.* [3] are considered the first to develop such a strategy. In this study, path-crossover operators were used, and the authors claim that their new strategy is a genetic algorithmic variant of the PR strategies initially proposed by Glover [38]. In our state-of-the-art comparison study, we name this method GGA to represent the Greedy Genetic Algorithm. The second method considered is the GRASP embedded PR algorithm developed by Oliveira *et al.* [71]. In this method, a few problems (eight problem instances from the QAPLIB) were considered and with run times reaching 25%, 50%, and 75% of the best known solution. We consider this method due to the emerging nature of GRASP and PR as new metaheuristic method. James *et al.* [45] proposed a sequential and parallel PR-based TS algorithm for QAP in 2005. They used more than 100 problem instances from the QAPLIB with a computation experiment that was quite comprehensive. For our comparison study, we utilized the sequential PR-based TS method since

comparing parallel implementations with the rest of the methods would not provide any fair results/conclusions. In the following tables (Tables 5.5-5.6), we abbreviate the sequential PR-based TS as SeqPR. Table 5.5 summarizes the findings for the 63 moderate problem classifications.

Table 5.5 Comparison with State-of-the-Art PR-Based Methods for the Moderate Problems (The Best Percentage Deviation Values from the Best Known Solutions are Boldfaced)

Problem name	GA-PM-PR	IA-PM-PR	SA-PM-PR	TS-PM-PR	SeqPR (James *et al.*,2005)	GRASP with PR (Oliveira *et al.*,2004)	GGA (Ahuja *et al.*, 2000)
Chr12a	**0.000%**	**0.000%**	**0.000%**	**0.000%**	-	-	**0.000%**
Chr12b	**0.000%**	**0.000%**	**0.000%**	**0.000%**	-	-	**0.000%**
Chr12c	**0.000%**	**0.000%**	**0.000%**	**0.000%**	-	-	**0.000%**
Chr18b	**0.000%**	**0.000%**	**0.000%**	**0.000%**	-	-	**0.000%**
Chr22a	1.917%	1.235%	12.541%	1.592%	-	75.000%	**0.750%**
Chr22b	0.581%	1.421%	10.268%	2.648%	-	-	**0.000%**
Els19	**0.000%**	**0.000%**	2.749%	**0.000%**	**0.000%**	-	**0.000%**
Esc32d	**0.000%**	**0.000%**	9.000%	**0.000%**	**0.000%**	-	**0.000%**
Esc32g	**0.000%**	**0.000%**	3.196%	**0.000%**	**0.000%**	-	**0.000%**
Esc64a	**0.000%**	**0.000%**	8.621%	**0.000%**	**0.000%**	-	**0.000%**
Kra30a	1.339%	1.350%	11.350%	0.175%	**0.000%**	75.000%	**0.000%**
Kra30b	0.252%	0.558%	12.000%	**0.000%**	-	-	**0.000%**
Kra32	1.421%	1.759%	12.356%	**0.000%**	-	-	-
Lipa20b	**0.000%**	**0.000%**	14.208%	1.472%	-	-	**0.000%**
Lipa30b	**0.000%**	**0.000%**	19.157%	1.145%	-	-	**0.000%**
Lipa40b	**0.000%**	3.763%	22.183%	1.015%	-	-	**0.000%**
Lipa50b	**0.000%**	16.033%	22.512%	0.806%	-	-	**0.000%**
Lipa60b	19.902%	20.320%	24.426%	0.771%	-	-	**0.000%**
Lipa70b	21.584%	21.369%	25.110%	0.700%	-	-	**0.000%**
Lipa80b	23.005%	22.457%	26.320%	0.643%	-	-	**0.000%**
Lipa90b	23.385%	22.464%	26.542%	**0.000%**	**0.000%**	-	**0.000%**
Nug20	**0.000%**	**0.000%**	2.879%	**0.000%**	-	-	**0.000%**
Nug21	**0.000%**	**0.000%**	3.774%	**0.000%**	-	-	**0.000%**
Nug22	**0.000%**	**0.000%**	4.616%	**0.000%**	-	-	**0.000%**
Nug24	**0.000%**	**0.000%**	5.791%	**0.000%**	-	-	**0.000%**
Nug25	0.107%	0.053%	5.342%	**0.000%**	-	-	**0.000%**
Nug27	**0.000%**	0.038%	5.808%	0.138%	-	-	**0.000%**
Nug30	1.078%	0.719%	8.328%	**0.000%**	-	- 75.000%	0.070%
Scr15	**0.000%**	**0.000%**	4.122%	**0.000%**	-	-	-
Scr20	0.025%	**0.000%**	5.777%	0.797%	-	-	**0.000%**
Sko42	2.277%	1.809%	11.447%	0.958%	**0.025%**	-	0.250%
Sko49	2.660%	2.035%	10.126%	0.784%	**0.071%**	-	0.210%
Sko56	2.856%	1.556%	10.639%	1.142%	**0.066%**	-	0.020%
Sko64	3.522%	2.693%	10.046%	0.954%	**0.054%**	-	0.220%
Sko72	3.514%	3.275%	10.327%	0.545%	**0.111%**	-	0.290%
Sko81	3.994%	3.350%	9.959%	0.808%	**0.060%**	-	0.200%
Sko90	4.163%	3.474%	10.851%	0.725%	**0.134%**	-	0.270%
Sko100a	4.808%	3.626%	10.109%	0.785%	**0.092%**	-	0.210%
Sko100b	4.495%	3.445%	10.748%	0.768%	-	-	0.140%
Sko100c	4.345%	3.763%	10.569%	1.122%	-	-	0.200%
Sko100d	3.665%	3.690%	10.222%	**0.861%**	-	-	0.170%

Sko100e	4.610%	3.830%	10.666%	1.066%	-	-	0.240%
Sko100f	4.414%	3.309%	8.614%	1.134%	-	-	0.290%
Tai20a	0.304%	**0.000%**	5.733%	**0.000%**	0.246%	-	-
Tai20b	**0.000%**	**0.000%**	3.188%	0.943%	**0.000%**	-	-
Tai25a	1.752%	0.642%	6.878%	**0.069%**	0.640%	-	-
Tai25b	**0.000%**	0.069%	5.801%	1.524%	**0.000%**	-	-
Tai30a	2.706%	1.758%	7.428%	**0.256%**	0.614%	-	-
Tai30b	0.149%	**0.000%**	5.784%	1.532%	**0.000%**	-	-
Tai35a	3.407%	2.675%	8.925%	0.529%	**0.109%**	-	-
Tai35b	0.427%	0.344%	8.594%	1.960%	**0.037%**	-	-
Tai40a	4.234%	3.583%	9.628%	1.054%	1.109%	-	-
Tai40b	0.575%	0.152%	13.166%	2.174%	**0.000%**	-	-
Tai50a	4.720%	4.580%	10.185%	**0.312%**	1.263%	-	-
Tai50b	2.647%	1.040%	18.461%	2.737%	**0.062%**	-	-
Tai60a	5.617%	5.004%	10.849%	**0.418%**	1.416%	-	-
Tai60b	2.197%	1.980%	25.966%	**0.000%**	0.042%	-	-
Tai80a	5.486%	4.797%	9.972%	1.623%	**1.109%**	-	-
Tai80b	6.815%	4.263%	22.417%	**2.457%**	-	-	-
Tai100b	5.370%	5.024%	9.172%	**1.096%**	-	-	-
Tai100a	5.909%	4.844%	23.007%	**0.524%**	0.966%	-	-
Tho30	0.604%	0.571%	9.405%	0.590%	-	75.000%	**0.000%**
Tho40	2.776%	2.415%	14.659%	0.324%	-	-	**0.320%**

For this comparison study, we selected the best implementations from each method. For example, for GA, the combination of PR with PM yielded the best results. Following this process, we presented GA-PM-PR, IA-PM-PR, SA-PM-PR, and TS-PM-PR for the analysis. From Table 5.4, we see that our implementations are as good as the state-of-the-art methods. Furthermore, our results are more robust, as the other studies did not replicate their experiments as frequently as we did (50 replications). GA-PM-PR, IA-PM-PR, and TS-PM-PR were found to have good solutions and are, thus, good competitors for SeqPR and GGA. In addition, TS-PM-PR was able to provide better solutions for the Kra32, Nug30, Sko100d, Tai20a, Tai25a, Tai30a, Tai50a, Tai60a, Tai60b, Tai80b, Tai100b, and Tai100a problems.

Table 5.6 depicts the same analysis carried out for the 14 hard problems. In this analysis, we compared our four methods with the three state-of-the-art methods in the literature. The SeqTS was taken out of the analysis since they did not consider these 14 problems for that specific computational study. Similar to the previous case,

GA-PM-PR, IA-PM-PR, and TS-PM-PR generated solutions that were as good as the other methods. TS-PM-PR was able to provide very competitive solutions for the Chr15a, Chr15b, Chr15c, Chr22c, Esc32a, Esc32b, Esc128, Ste36a, Ste36b, and Ste36c problem instances.

When the run times are considered, all the four of the methods recorded quite impressive times since, except for the POS_PR implementation procedures, the methods do not have any initialization heuristics, greedy speedups, multi-start initializations, or other adaptive techniques embedded in their individual main programs.

Table 5.6 Comparison with State-of-the-Art PR-Based Methods for the Hard Problems (The Best Percentage Deviation Values from the Best Known Solutions are Boldfaced)

Problem name	GA-PM-PR	IA-PM-PR	SA-PM-PR	TS-PM-PR	GRASP with PR (Oliverira *et al.*,2004)	GGA (Ahuja *et al.*, 2000)
Chr15a	**0.000%**	**0.000%**	1.152%	**0.000%**	-	-
Chr15b	**0.000%**	**0.000%**	**0.000%**	**0.000%**	-	-
Chr15c	**0.000%**	**0.000%**	16.372%	**0.000%**	-	-
Chr18a	**0.000%**	**0.000%**	33.646%	3.586%	-	-
Chr20a	1.460%	**0.000%**	29.562%	0.183%	-	-
Chr20b	5.048%	6.440%	21.323%	5.396%	-	-
Chr20c	**0.000%**	**0.000%**	15.953%	4.724%	-	-
Chr22c	12.698%	8.377%	69.442%	**9.747%**	-	-
Esc32a	4.615%	3.077%	46.154%	**0.000%**	-	**0.000%**
Esc32b	9.524%	**0.000%**	45.238%	**0.000%**	-	**0.000%**
Esc128	6.250%	15.625%	96.875%	**0.000%**	-	**0.000%**
Ste36a	2.645%	1.953%	35.482%	**0.000%**	75.000%	0.270%
Ste36b	2.183%	3.104%	57.797%	**0.143%**	-	-
Ste36c	1.290%	1.792%	30.846%	**0.000%**	-	-

Compared to the PR-based methods discussed earlier, we carried out a detailed state-of-the-art comparison analysis for the six metaheuristics mentioned in the literature. These methods include the multi-start diversified TS (DivTS) by James *et al.*, 2009 [47]; the Robust Tabu Search (RTS) by Taillard, 1991 [84]; the GRASP by Pardalos *et al.*, 1994 [56]; the ant colony-based local search algorithm (ACO-GL/LS)

by Tseng and Liang, 2006 [87]; the integrated GA-based TS (GA/TS) by Misevičius, 2004 [69]; and the local search metaheuristic (ILS3) by Stützle, 2006 [83]. These methods have been widely cited by the literature to be very competitive in solving QAPs.

In Table 5.7, we present the 47 QAPs extracted from the QAPLIB that stem from real-life problems. Comparing the performance of real-life problems allows us to generate insights into how our best performing method (TS-PM-PR) would react to general real-life problems, which will enable practitioners to weigh the pros and cons of the method.

Table 5.7 Comparison with Long-Run Solutions of State-of-the-Art Metaheuristics for Real-Life Problems (The Best Percentage Deviation Values from the Best Known Solutions are Boldfaced)

*DivTS: Multi-Start Diversified TS (James *et al.*, 2009) [47], RTS: Robust Tabu Search (Taillard, 1991) [84], GRASP (Pardalos *et al.*, 1994) [56], ACO-GL/LS (Tseng and Liang, 2006) [87], GA/TS (Misevičius, 2004) [69], and ILS3 (Stützle, 2006) [83]

Problem name	DivTS*	RTS*	GRASP*	ACO-GL/LS*	GA/TS*	ILS3*	TS-PM-PR	TS-PM-PR (Run Time in Sec)
Bur26a	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	24.203
Bur26b	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	24.765
Bur26c	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	33.188
Bur26d	**0.000%**	**0.000%**	**0.000%**	**0.000%**			0.001%	41.766
Bur26e	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	25.031
Bur26f	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	44.265
Bur26g	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	25.015
Bur26h	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	36.500
Chr12a	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	12.296
Chr12b	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	21.156
Chr12c	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	16.187
Chr15a	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	28.312
Chr15b	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	14.610
Chr15c	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	18.859
Chr18a	**0.000%**	**0.000%**	**0.000%**	**0.000%**			3.586%	16.703
Chr18b	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	32.031
Chr20a	**0.000%**	**0.000%**	**0.000%**	**0.000%**			0.183%	22.218
Chr20b	**0.200%**	0.400%	3.133%	0.870%			5.396%	22.859
Chr20c	**0.000%**	**0.000%**	**0.000%**	**0.000%**			4.724%	59.156
Chr22a	**0.000%**	**0.000%**	0.552%	**0.000%**			1.592%	39.953
Chr22b	**0.152%**	0.213%	1.421%	**0.000%**			2.648%	30.718
Chr25a	0.943%	0.601%	4.636%	**0.000%**			9.747%	49.203
Kra30a	**0.000%**	**0.000%**	**0.000%**	**0.000%**	**0.000%**		0.175%	32.906
Kra30b	**0.000%**	**0.000%**	**0.000%**	**0.000%**	0.089 %		**0.000%**	36.734
Kra32	**0.000%**	**0.000%**		**0.000%**	0.019%		**0.000%**	44.344
Ste36a	**0.000%**	**0.000%**	0.651%	**0.000%**			**0.000%**	49.031

Ste36b	**0.000%**	**0.000%**	**0.000%**	**0.000%**	0.149%		0.143%	48.890
Ste36c	**0.000%**	**0.000%**	0.188%	**0.000%**	**0.000%**		**0.000%**	46.734
Esc16a	**0.000%**	**0.000%**	**0.000%**	**0.000%**	0.066%		**0.000%**	42.750
Esc16b	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	15.111
Esc16c	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	16.546
Esc16d	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	18.437
Esc16e	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	16.250
Esc16f	**0.000%**	**0.000%**	**0.000%**	**0.000%**		9.521%	**0.000%**	25.200
Esc16g	**0.000%**	**0.000%**	**0.000%**	**0.000%**		**0.000%**	**0.000%**	25.296
Esc16h	**0.000%**	**0.000%**	**0.000%**	**0.000%**		0.051%	**0.000%**	22.218
Esc16i	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	23.656
Esc16j	**0.000%**	**0.000%**	**0.000%**	**0.000%**			**0.000%**	23.703
Esc32a	**0.000%**	**0.000%**	**0.000%**	**0.000%**		0.227%	**0.000%**	41.406
Esc32b	**0.000%**	**0.000%**	**0.000%**	**0.000%**	**0.000%**	**0.000%**	**0.000%**	44.860
Esc32c	**0.000%**	**0.000%**	**0.000%**	**0.000%**	**0.000%**		**0.000%**	41.203
Esc32d	**0.000%**	**0.000%**	**0.000%**	**0.000%**	**0.000%**		**0.000%**	34.703
Esc32e	**0.000%**	**0.000%**	**0.000%**	**0.000%**	**0.000%**		**0.000%**	32.671
Esc32f	**0.000%**	**0.000%**	**0.000%**	**0.000%**	**0.000%**		**0.000%**	33.593
Esc32g	**0.000%**	**0.000%**	**0.000%**	**0.000%**	**0.000%**		**0.000%**	299.410
Esc64a	**0.000%**	**0.000%**	**0.000%**	**0.000%**	**0.000%**		**0.000%**	201.670
Esc128	**0.000%**	**0.000%**	**0.000%**	**0.000%**	**0.000%**		**0.000%**	1222.100

Table 5.6 depicts the long-run percentage deviations from the best known values of the six metaheuristics compared with TS-PM-PR. The last column contains information on the run times of our method (the computational configurations include Microsoft Windows XP, Intel® Pentium® 4 CPU 3.00 GHz 2.99 GHz, and 1.00 GB of RAM). The solutions recorded for each method are from James *et al.*, 2009 [47], and the values are presented for longer running times. Therefore, our method was able to generate good solutions within very reasonable time frames. Hence, our solutions have proven to be a very competitive metaheuristic.

5.8 Conclusions

In conclusion, this chapter investigated the effects of POS_PR augmentation on the performance of each tuned path-based and population-based metaheuristic. Based on the results obtained, we carried out separate analyses for each representation method. Furthermore, detailed analyses were performed to demonstrate how the previous classifications of problems (moderate vs. hard problems) were improved due

to our new diversification mechanisms. These results generated some interesting findings. To begin with, the GAs, IAs, and TSs were significantly improved irrespective of the representation schemes used. Many of the moderate problems can now be re-classified as trivial, while many of the hard problems can be regrouped as either trivial or moderate problems. One interesting observation was recorded regarding the performance of the SAs. They were not improved by the POS_PR augmentation for either representation scheme, and most often, the augmentation deteriorated the quality of the solutions. Some possible reasons for this phenomenon to occur can be highlighted based on fitness evaluations and associated randomness when creating solutions. When random keys representation was used for path-based methods to create initial solutions and for the POS_PR solution combination procedures, the associated randomness is higher compared to permutation encoding scheme. Because the number of solutions evaluated at a given time is fewer than the population-based methods. This causes the deterioration of the performance and permutation representation has less randomness allied; hence, creates more successful solutions from POS_PR augmentation.

The comparison analysis followed with CART classifications to identify the splitting criterion. Representation scheme played an important role in determining the quality of a particular solution. From the CART analysis figures, it was found that the best performing methods are common for the hard and moderate problems. However, when attention was given to problem characteristics, population-based methods were more susceptible to the sparsity of the data matrices; whereas, path-based metaheuristics were dominated or controlled by distance dominance and the size of the problem as opposed to by the sparisity of the matrices. In order to see the positive

effects of POS_PR for path-based metaheuristics, permutation-based representation schemes should be implemented.

This chapter concluded with a comparison study carried out with a few state-of-the-art PR-based diversification methods. These included the Greedy Genetic Algorithm (GGA) [3], GRASP with PR [71], and Sequential PR (SeqPR) [45] from the literature. The results yielded that the POS_PR-augmented methods were as competitive as these methods and that TS-PM-PR generated competitive solutions for quite a number of the problem instances. Finally, using six competitive metaheuristics and 47 real-world problems, we carried out a detailed comparison of solution quality. The winning TS-PM-PR seems to be very competitive in terms of solution quality as well as run time compared to the other state-of-the-art metaheuristics.

This study lays the foundation for many research studies that can focus on different variants of PR mechanisms to generate better solutions irrespective of problem characteristics or representation schemes.

CHAPTER SIX

6. CONCLUDING REMARKS

6.1. Introduction

In this concluding chapter, we compare the overall results of the preceding chapters, present a discussion of the results, and provide directions for future research. The chapter is organized as follows. We begin by summarizing the results generated by the parameter tuning procedures discussed in chapter 3. This section is followed by the effects of the solution representations and problem characteristics on the performance of metaheuristics, as presented and discussed in chapter 4. Next, we discuss the importance of PR augmentation, which was presented in chapter 5, and then summarize the overall findings of the dissertation. Finally, we present some future directions for research on fitness landscape analysis and conclude the chapter with a discussion of our results.

6.2. Effects of Parameter Tuning on Performance

An extensive parameter tuning procedure was carried out for the four selected metaheuristics to investigate the effects of parameter tuning on performance. Using a full factorial design framework, we identified metaheuristic design factors for each method by replicating each treatment in a high throughput computing environment. It was found that parameter tuning does significantly improve the quality of the solutions for both the path-based metaheuristics and population-based metaheuristics. Using parametric and non-parametric statistical analysis tools, we tested the statistical significance of the results. We also found that some metaheuristic design factors are less sensitive to parameter changes than others. Specifically,

cooling schedule for SA, neighborhood size for TS, mutation probability for GA, and affinity adjustments for IA were less responsive to parameter tuning. Further, some methods showed significant interaction effects between two metaheuristic design factors. Specifically, the size of the neighborhood and the size of the tabu list for TS and affinity threshold and affinity adjustment for IA.

6.3. Effects of Solution Representation on Performance

During the parameter tuning process, we investigated how the performance of a particular metaheuristic is affected by the solution representation scheme. We found that permutation-based representation schemes are affected less by tuning than by the random keys. This phenomenon was significantly more visible for the path-based metaheuristics than for the population-based metaheuristic methods. One interesting observation was irrespective of the philosophical differences (whether it is a path-based method or a population-based method), permutation representation schemes outperformed random keys representations for many QAP instances. Even though random keys representations are widely applied in other combinatorial optimization problems, such as scheduling, the permutation representation is more suitable for cases like QAPs.

Based on the results of the parameter tuning procedure and the performance of each method for each representation scheme, 130 QAPLIB problems were categorized into three categories: trivial problems, moderately difficult problems, and hard problems. A comparison study was carried out using four problem classes from the literature, and the tuned metaheuristics were evaluated in each of these categories. The extensive parameter tuning procedure generated competitive solutions for certain

QAPs, and the representation scheme played an important role in determining solution quality.

Overall, TS with both representation schemes and IA with permutation representation outperform the rest of the metaheuristics investigated in this dissertation. In chapter 3, we concluded the comparison study by carrying out a comparison study with the greedy GA initially developed by Ahuja *et al*.[3]. The underlying results of this comparison study indicated that the tuned GA developed here outperformed the greedy GA for the selected problem instances.

6.4. Effects of Problem Characteristics on Performance

In chapter 4, the moderate and the hard problems that were classified in chapter 3 were carefully analyzed using in-depth problem characteristics. These measures included problem size, flow and distance dominance measures, sparsity, and coefficient of correlation measures. In order to compare and contrast each population-based method with the path-based methods, we used Classification and Regression Tree (CART) tools. The moderately difficult and hard problems identified in the chapter 3 were considered for these analyses. We found that the sparsity of the data matrices, flow dominance, and the size of the problem play an important role in determining a particular metaheuristic's performance.

The findings from chapter 4 raised the following research questions: (1) are path-based metaheuristic approaches more suitable for QAPs? (2) Are population-based metaheuristics unsuitable for these problems? and (3) when deciding on the most suitable representation scheme, are permutation representations more suitable than random keys representations? In order to answer these questions, we carried out

a search trajectory analysis for all of the hard and the moderate problems (77 input files) and recorded the best solution for each method found along the search path.

The results of the overall analysis suggest that TS and IA-PM have built-in diversification and intensification mechanisms unlike the other algorithms and that this feature appears to make TS and IA-PM very effective for QAPs. GAs and IAs use their population-based nature and begin the search trajectory with more competitive solutions than the SAs and TSs. As the number of fitness evaluations increase, the solution quality of TS dominates the other methods, and the deviation from the best known solution drops more sharply than the rest. The TSs and IA-PM show less stagnation than the other methods.

The findings of this chapter lay the foundation for further research studies on improving individual search trajectories for less intelligent metaheuristics. In chapter 5, a diversification method based on PR was proposed to improve the solution quality of the less diversified metaheuristics, by expanding their search space.

6.5. Effects of PR Mechanisms on Performance

In chapter 5, we implemented POS_PR mechanisms for the two representation schemes. Two comparison studies were carried out for the two representation schemes.

In the random keys POS_PR comparison study, we found that for all of the moderate and hard problem instances, GA and IA showed positive improvement. Considering the overall performance, POS_PR can be considered a promising method for improving random keys population-based metaheuristics. In the case of path-based metaheuristics, the SAs' and TSs' performances were adversely affected by the

augmentation. Except for a few problem instances, the recorded performances were negative. Hence, POS_PR is not very suitable for random keys-based SA/TS in the QAP context.

In the comparison study for permutation-based metaheuristics, the performances of the tuned population-based methods (GA-PM and IA-PM) were compared with their POS_PR augmentations.

In the comparison study for the permutation-based metaheuristics, the performances of the tuned population-based methods (GA-PM and IA-PM) and the two path-based methods (TS-PM and SA-PM) were compared with their POS_PR augmentations. It was evident that from the 77 problem instances considered, SA yielded marginal returns irrespective of the representation scheme. In the path-based analysis, we found that POS_PR implementations are more beneficial when permutation representations are used rather than random keys. Random keys GA and IA were significantly improved by POS_PR and POS_PR, and the augmented TS-PM outperformed the rest with more than 90% improvement in solution quality.

From the CART analysis, we found that the best performing methods are the same for both the hard and moderate problems. In such cases, IAPM, TSPM, IAPM-PR, and TSPM-PR are the most competitive metaheuristics. However, when problem characteristics are considered for population-based methods, the sparsity of the data matrices determines whether a particular problem is considered a moderately difficult problem or a hard problem to solve. For path-based metaheuristics, distance dominance and the size of the problem play a bigger role than the sparsity of the matrices.

To conclude the analysis, we compared the best performing metaheuristics found (TSPM-PR) with other state-of-the-art PR-based metaheuristics discussed in the literature (SeqPR, GRASP with PR, and GGA). We found that TSPM-PR was able to find better solutions to 19 problem instances, thus proving it to be superior to the other PR-based methods proposed in the literature. In order to conclude the chapter, we consider six more state-of-the-art metaheuristics mentioned in the literature to solve the QAPs (DivTS: Multi-Start Diversified TS (James et al., 2009) [47], RTS: Robust Tabu Search (Taillard, 1991) [84], GRASP (Pardalos et al., 1994) [56], ACO-GL/LS (Tseng and Liang, 2006) [87], GA/TS (Misevičius, 2004) [69], and ILS3(Stützle,2006) [83]), comparing their solutions with ours. Compared to the long-run solutions obtained by these methods, TSPM-PR did very well, recording fairly short run times. Hence, this method is a very competitive PR-based metaheuristic.

6.6. Representation Scheme, Parameter Tuning, and PR Augmentation

We utilized three approaches in this dissertation to improve the solution quality of the two selected population-based metaheuristics and the two path-based metaheuristics. Namely, we investigated the following:

- Effects of representation scheme
- Effects parameter tuning
- Effects of augmenting a diversification method

In each chapter, we saw how these three procedures affected the performance of each metaheuristic. We found that irrespective of the philosophical differences (whether the method is a path-based metaheuristic or a population-based metaheuristic), all four methods benefited from permutation-based representation

schemes. It can be concluded that for the selected problem domain, random keys representation is not very suitable. However, once the parameter tuning process was carried out, the random keys-based methods showed significant improvement, while the permutation-based metaheuristics showed less improvement, meaning that the representation scheme itself was more robust to parameter changes. From the overall analysis, TS was found to be more robust out of the other three methods, and GA and IA were significantly improved by the parameter tuning procedures.

When PR augmentation was considered, we saw some significant improvement in solution quality in TS, GA, and IA as opposed to SA. SA's performance was least affected by the representation scheme, and the PR augmentation caused it to create worse solutions. GA was significantly improved by the PR augmentation irrespective of the representation scheme. PR augmented permutation-based TS outperformed the rest. We found this metaheuristic to be very competitive compared to the rest of the PR-based methods in the literature, and when compared to other state-of-the-art methods, it generated fairly good solutions within a very quick time span.

Based on the results of the three chapters, the 130 input files of the QAPLIB can be reorganized based on the appropriateness of the metaheuristics capable of solving them. The following Table 6.1 depicts Bur* files of the QAPLIB using possible rankings of metaheuristics matched with problem characteristics.

Name	Ranking of Metaheuristics (Based on the Average Deviation from the Best-Known)								FD	DD	Sparsity	Correlation N: Negative P: Positive	Problem Size
	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	Rank 8					
Bur26a	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	HIGH	LOW	LOW	LOW (N)	LOW
Bur26b	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	HIGH	LOW	LOW	LOW (N)	LOW
Bur26c	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	HIGH	LOW	LOW	LOW (N)	LOW
Bur26d	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	HIGH	LOW	LOW	LOW (P)	LOW
Bur26e	TS-PM-PR	GA-PM-PR	IA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	HIGH	LOW	LOW	LOW (P)	LOW
Bur26f	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	HIGH	LOW	LOW	LOW (N)	LOW
Bur26g	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	HIGH	LOW	LOW	LOW (P)	LOW
Bur26h	TS-PM-PR	GA-PM-PR	IA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	HIGH	LOW	LOW	LOW (N)	LOW
Bur26	**ALL**	**ALL**	**ALL**	**ALL**	**ALL**	**ALL**	**ALL**	**ALL**	LOW	LOW	LOW	LOW (P)	LOW

Table 6.1: Classification of Bur* Files of QAPLIB with POS_PR Augmentation and Problem Characteristics
Flow Dominance (FD): LOW (<=60), Moderate (60-200), and High (200>)
Distance Dominance (DD): LOW (<=100), Moderate (100-200), and High (200>)
Sparsity: LOW (<=0.5), Moderate (0.5-0.7), and High (0.7>)
Coefficient of Correlation: LOW (<=∓0.1), Moderate (∓0.1 - ∓ 0.5), and High (∓>0.5)
Problem Size: LOW (<=40), Moderate (40-75), and High (75>)

The rest of the input files are presented in Appendix G and from Table 6.1 it is clearly visible that for Bur* instances, TS-PM-PR is the most appropriate method. These files consist of high FD values but low in rest of the problem characteristics. It is interesting to note that the same TS with RK (TS-RK-PR) performs worse, comparatively to the other methods for the same set of problems. Hence, once again highlights the importance of using the most appropriate solution encoding procedures.

With these findings at hand, we present some future research directions and possible extensions to this dissertation in the next section.

6.7. Fitness Landscape Analysis

As a possible future extension for this dissertation, we propose a fitness landscape analysis study. In the previous chapters, we investigated how search trajectories change when representation schemes or diversification mechanisms change. We clearly saw that search trajectories shifted downward (resulting in low fitness function values) with improvements in the number of solutions found or the ability to find more diversified solutions. To illustrate this point, Figures 6.1- 6.3 were created using three input files in order to compare how the exploration of the search spaces improved by POS_PR augmentation.

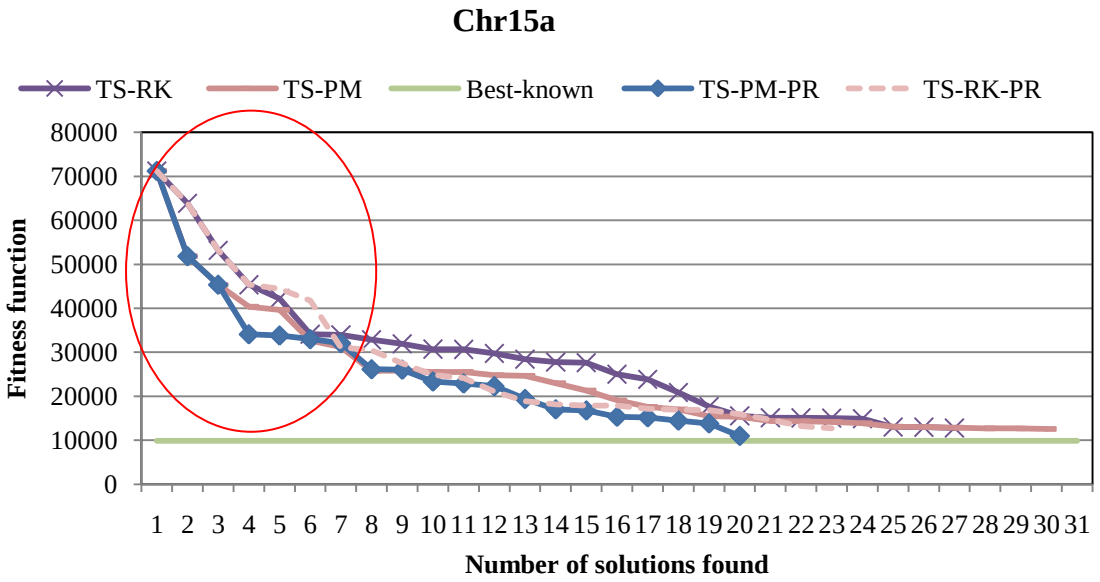


Figure 6.1: Search Trajectory Comparison of TS for Chr15a

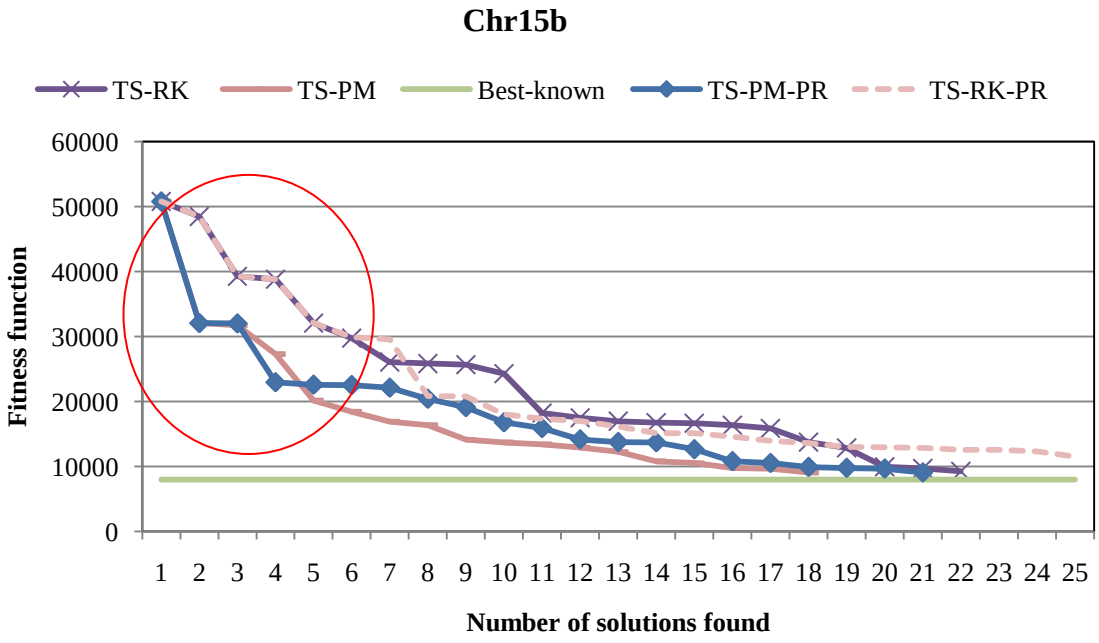


Figure 6.2: Search Trajectory Comparison of TS for Chr15b

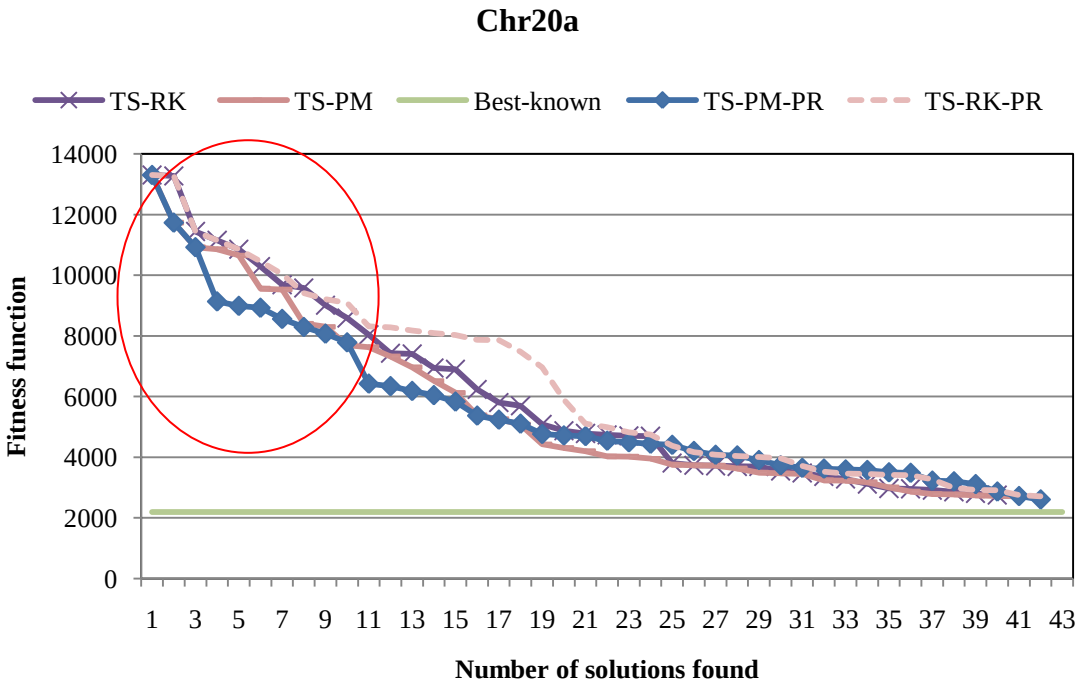


Figure 6.3: Search Trajectory Comparison of TS for Chr20a

Using these three figures, we can see that TS-PM-PR found the best known solutions with fewer solutions and minimal stagnation in the search trajectory. The diversified areas are highlighted using circles, and it is clearly evident that POS_PR augmentation creates an efficient beginning for each search path.

The search trajectory of a particular metaheuristic has to be closely matched to the actual fitness landscape of a problem. Thus, when designing metaheuristics, we need to minimize the gap in the actual fitness landscape and the shape of the search trajectory.

Future research can be done in this area to actually model the search trajectory of particular metaheuristic using different modeling techniques. Using different

parameter tuning procedures, we can predict how closely we can mimic the actual fitness landscape.

In this endeavor, researchers could investigate the Fitness Distance Correlation (FDC), which measures the extent to which the fitness function values are correlated with the distance to a global optimum. These measures and other modeling/ statistical analysis techniques can be used to carefully project a search trajectory that can be closely superimposed onto the actual fitness landscape of any combinatorial optimization problem.

____END____

LIST OF REFERENCES

[1]. Abramson N (1963). Information Theory and Coding. McGraw Hill: New York.

[2]. Adenso-Daz B and Laguna M (2006). Fine-tuning of algorithms using fractional experimental design and local search. Operations Research 54(1): 99-114.

[3]. Ahuja R K (2000). A greedy genetic algorithm for the quadratic assignment problem. Computers & Operations Research 27: 917-934.

[4]. Alfandari L, Plateau A and Tolla P (2001). A two-phase path-relinking algorithm for the Generalized Assignment Problem, In: Proceedings of the fourth Metaheuristics International Conference of Porto, Portugal

[5]. Anstreicher K, Brixius N, Goux J P and Linderoth J (2000). Solving large quadratic assignment problems on computational Grids. Mathematical Programming 91 (3): 563-588.

[6]. Anstreicher K M (2003).Recent Advances in the Solution of Quadratic Assignment Problems. Mathematical Programming 97(1-2):27-42.

[7]. Arkin E M , Hassin R , Sviridenko M (2001).Approximating the maximum quadratic assignment problem. Information Processing Letters 77(1):13–16

[8]. Arostegui Jr. M A, Kadipasaoglu S N, and Khumawala B M (2006). An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems. International Journal of Production Economics 103(2):742–754.

[9]. Bachelet V, Preux P and Talbi E G (1996). Parallel Meta-Heuristics: Application to the Quadratic Assignment Problem. In: Proceedings of the Parallel Optimization Colloquium, (Versailles, France).

[10]. Baños R , Gil C, Reca J and Martínez J (2007). Implementation of scatter search for multi-objective optimization: a comparative study. Computational Optimization and Applications. Springer: Netherlands.

[11]. Barr R, Golden B, Kelly J, Resende M and Stewart W (1995). Designing and Reporting on Computational Experiments with Heuristic Methods. Journal of Heuristics 1: 9–32.

[12]. Battiti R, Tecchiolli G (1994). Simulated Annealing and Tabu Search in the Long run: A comparison on QAP Tasks. Computers Mathematical Applications 28:1-8.

[13]. Bean J C (1994). Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing 6:154–160

[14]. Beausoleil R P and Montejo R A (2009).A Study with Neighborhood Searches to Deal With Multiobjective Unconstrained Permutation Problems. Journal of Industrial and Management Optimization.5(2):193-216

[15]. Bersini H and Varela F J (1991). The immune recruitment mechanism: A selective evolutionary strategy. In Proc. Fourth Int. Conf Genetic Algorithms San Mateo, CA, USA: Morgan Kaufmann 520-526.

[16]. Birattari M (2005). The Problem of Tuning Metaheuristics as Seen from a Machine Learning Perspective. Ph.D. thesis, vol.292 of Dissertationen zur Künstlichen Intelligenz. Akademische Verlagsgesellschaft Aka GmbH, Berlin, Germany.
[17]. Burkard, R. E. (1984). Quadratic Assignment Problems. European Journal of Operational Research, 15(3), 283-289.
[18]. Burkard R E, Karisch, S E, Rendl F (1997).QAPLIB – A Quadratic Assignment Problem Library. Journal of Global Optimization 10 (4): 391–403.
[19]. Chinchuluun A and Pardalos P (2007). A survey of recent developments in multiobjective optimization. Annals of Operations Research.154(1):29-50
[20]. Coy S P, Golden, B L, Runger, G C, Wasil, E A (2001). Using experimental Design to Find Effective Parameter Setting for Heuristics. Journal of Heuristics 7(1):77-97
[21]. Cung V, Mautor T, Michelon P and Tavares A (1997). A scatter search approach for the quadratic assignment problem. In: Proceedings of IEEE International Conference on Evolutionary Computation and Evolutionary Programming: Indianapolis, United States of America, 165-170.
[22]. Day R O and Lamont G B (2005). Multiobjective quadratic assignment problem solved by an explicit building block search algorithm - MOMGA-IIa.
[23]. Deb K and Agrawal S (1999). Understanding Interactions among Genetic Algorithm Parameters. In Proc. of Fifth Workshop on Foundations of Genetic Algorithms (FOGA), Madison, WI, USA 265-286.
[24]. Deb K (1999). An introduction to genetic algorithms. Sadhana-Academy Proceedings in Engineering Sciences, *24*, 293-315.
[25]. Deb K (2000). An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering, *186*(2-4), 311-338.
[26]. Deb K(2001).Multi-Objective Optimization Using Evolutionary Algorithms: Wiley
[27]. Deb K , Anand A and Joshi D (2002). A computationally efficient evolutionary algorithm for real-parameter optimization. Evolutionary Computation, *10*(4): 371-395.
[28]. Deb K, Pratap A, Agarwal S and Meyarivan T (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, *6*(2): 182-197.
[29]. Deineko V G , Woeginger G J, (2000). A study of exponential neighborhoods for the traveling salesman problem and for the quadratic assignment problem. Mathematical Programming, Series A 78: 519–542.
[30]. Drenzner Z (2008). Extensive Experiments with Hybrid Genetic Algorithm for the Solution of the Quadratic Assignment Problem. Computers and Operations Research.35:717-736.

[31]. Dreo J, Petrowski A, Siarry P and Taillard E (2005). Metaheuristics for Hard Optimization. Springer: New York.
[32]. Duman E and Or I (2007). The quadratic assignment problem in the context of the printed circuit board assembly process. Computers & Operations Research. 34(1), 163-179.
[33]. Ehrgott M(2005). Multicriteria Optimization. Birkhäuser :Berlin
[34]. Festa P (2002) Greedy Randomized Adaptive Search Procedures, Met heuristics. 7(4)
[35]. Finke G, Burkard R E and Rendl F (1987).Quadratic Assignment Problems. Annals of Discrete Mathematics.31:61–82
[36]. Gambardella L M, Taillard É and Dorigo M (1999) Ant Colonies for the Quadratic Assignment Problem, The Journal Of The Operations Research Society, 50(2):167-176.
[37]. Glover F (1989).Tabu Search Part I. ORSA Journal on Computing 1(3):190-206.
[38]. Glover F (1998). A Template for Scatter Search and Path Relinking. In: Hao J K, Lutton E, Ronald E, Schoenauer M and Snyers D (Eds.). Artificial Evolution Lecture Notes in Computer Science 1998. Springer, pp. 3-51.
[39]. Glover F, Laguna M and Marti R (2000). Fundamentals of scatter search and path relinking .Control and Cybernetics. 29(3):653-684.
[40]. Glover F, Kochenberger G A, EDS. (2003). Handbook of Metaheuristics. Kluwer Academic Publishers: Dordrecht, The Netherlands
[41]. Glover F, Laguna M and Marti R (2005). New ideas and applications of scatter search and path relinking. Lecture Notes in Computer Science. 3410,443-458.
[42]. Glover F, Laguna M and Marti R (2006). Principles of scatter search, European journal of operations research, 169, pp 359-372.
[43]. Hansen P, Lih K W (1992). Improved algorithms for partitioning problems in parallel, pipelined, and distributed computing. IEEE Transactions on Computers 41(6):769–771
[44]. Holland J H (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press: MI.
[45]. James T, Rego C and Glover F (2005). Sequential and parallel path-relinking algorithms for the quadratic assignment problem. IEEE Intelligent Systems 20 (4): 58–65.
[46]. James T, Rego C and Glover F (2009). A cooperative parallel tabu search algorithm for the quadratic assignment problem. European Journal of Operational Research 195:810-826
[47]. James T, Rego C, Glover F (2009). Multistart Tabu Search and Diversification Strategies for the Quadratic Assignment Problem. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions. 39(3):579-596

[48]. Jones D F, Mirrazavi S K, and Tamiz M (2002). Multiobjective meta-heuristics: an overview of the current state-of-the-art, European Journal of Operational Research 137(1):1-9

[49]. Kelly J P, Laguna M, Glover F (1994). A study of diversification strategies for the quadratic assignment problem. Computers and Operations Research 21(8):885-893

[50]. Kirkpatrick S G, Delatt Jr. C D and Vecchi M P (1983).Optimization by Simulated Annealing. Science 220: 671–680.

[51]. Kleeman M P, Day R O and Lamont G B (2004). Analysis of a parallel MOEA solving the multi-objective quadratic assignment problem.

[52]. Knowles J D and Corne D W (1996) .Towards landscape analyses to inform the design of a hybrid local search for the multiobjective quadratic assignment problem. In: Proceedings of the Parallel Optimization Colloquium, (Versailles, France).

[53]. Knowles J and Corne D (2003). Instance generators and test suites for the multiobjective quadratic assignment problem.

[54]. Koopmans T C and Beckmann M J (1957). Assignment problems and the location of economic activities. Econometrica 25: 53–76.

[55]. Li D and Landa-Silva D (2009). An Elitist GRASP Metaheuristic for the Multi-objective Quadratic Assignment Problem. Evolutionary Multi-Criterion Optimization.5467:481-491

[56]. Li D, Pardalos P M, and Resende M G C (1994). A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem. DIMACS series on Discrete Mathematics and Theoretical Computer Science, vol 16, P M Pardalos and H Wolkowicz,Eds. Baltimore ,MD, pp 237-261.

[57]. Litzkow M, Livny M and Mutka M (1988). Condor: A hunter of idle workstations In Proc. 8th Intl Conf. on Distributed Computing Systems. pp. 104-111.

[58]. Loiola E M , de Abreu N M M, Boaventura-Netto P O, Hahn P and Querido T (2007).A Survey For The Quadratic Assignment Problem. European Journal of Operational Research 176:657–690.

[59]. Lopez-Ibanez M, Paquete L and Stützle T (2004). On the design of ACO for the biobjective quadratic assignment problem.

[60]. Malakooti B and Dsouza G I (1987). Multiple Objective Programming for the Quadratic Assignment Problem. International Journal of Production Research. 25(2): 285-300.

[61]. Maniezzo V, Dorigo M and Colorni A (1995). Algodesk - An Experimental Comparison of 8 Evolutionary Heuristics Applied To the Quadratic Assignment Problem. European Journal of Operational Research 81:188-204.

[62]. Matsumoto M and Nishimura T (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Transactions on Modeling and Computer Simulation (TOMACS) 8(1):3-30
[63]. Mautor T and Roucairol C (1994). A new exact algorithm for the solution of Quadratic Assignment Problems. Discrete Applied Mathamatics.55 (3):281–293
[64]. Merz P and Freisleben B (1999). A comparison of Memetic Algorithms, Tabu Search, and Ant Colonies for the Quadratic Assignment Problem. In: Angeline P (eds).Congress on Evolutionary Computation (CES'99). IEEE Press. 2063-2070.
[65]. Merz P and Freisleben B. (2000) Fitness landscape analysis and memetic algorithms for the quadratic assignment problem, IEEE Trans. Evol. Comput., 4: 337–352.
[66]. Misevičius A (2005). A Fast hybrid Genetic Algorithm for the Quadratic Assignment Problem, Computational Optimization and Applications,30:95–111.
[67]. Misevičius A and Kilda B (2005). Comparison of Crossover Operators for the Quadratic Assignment Problem. Information Technology and Control. 34(2):109-119
[68]. Misevičius A (2006). A Tabu Search Algorithm for the Quadratic Assignment Problem, In: Proceedings of GECCO'06.
[69]. Misevičius A (2006). An improved hybrid genetic algorithm: New results for the Quadratic Assignment Problem. Knowledge –Based Systems. Vol 17 (2-4). Pp 65-73.
[70]. Nebro A J, Luna F, Alba E, Behamb A and Dorronsoro B (2006). AbYSS: Adapting Scatter Search for Multiobjective Optimization, Tech-Report: ITI-2006-2. Elsevier Science: 20.
[71]. Oliveira C A S, Pardalos P M and Resende M G C (2004).GRASP with Path-Relinking for the Quadratic Assignment Problem. Lecture Notes in Computer Science.3059:356-368
[72]. Pardalos P M and Grouse J V (1989). A Parallel Algorithm for the Quadratic Assignment Problem, In: Conference on High Performance Networking and Computing Proceedings of the 1989 ACM/IEEE conference on Supercomputing.
[73]. Park M W and Kim Y D (1998). A Systematic Procedure for Setting Parameters in Simulated Annealing Algorithms. Computers & Operations Research 24(3): 207–217.
[74]. Pierskalla W P (1967). The tri-substitution method for the three-multidimensional assignment problem. Canadian Operational Research Society Journal 5(2):71–81
[75]. Raidl G R (2006) .A Unified View on Hybrid Metaheuristics, Lecture Notes in Computer Science, 4030, 1611-3349.
[76]. Rardin R L and Uzsoy R (2001).Experimental Evaluation of Heuristic Optimization Algorithms: A Tutorial. Journal of Heuristics. 7(3):261-304

[77]. Rasmussen R (2007). QAP - not so hard in spreadsheets. Omega-International Journal of Management Science. 35(5):541-552.
[78]. Roy, Alain (2006).Introduction to Condor. http://www.nanohub.org/resources/1208/, accessed 20 June 2009.
[79]. Sahni S, Gonzales T (1976). P-complete approximation problems. Journal of the Association for Computing Machinery 23:555–565
[80]. Sinclair M (1993). Comparison of the performance of modern heuristics for combinatorial optimization on real data. Computers & Operations Research 20 (7): 687–695.
[81]. Steinberg, L (1961) .The backboard wiring problem: A placement algorithm. SIAM Review 3:37–50
[82]. Stützle T and Dorigo M (1999). Local Search and Metaheuristics for the Quadratic Assignment Problem. Metaheuristic network training project.
[83]. Stützle T (2006). Iterated local search for the quadratic assignment problem. European Journal of Operational Research. 174(3): 1519-1539.
[84]. Taillard E (1991). Robust Tabu search for the Quadratic Assignment Problem. Parallet Computing. vol 17, pp 443-455.
[85]. Tate, D. M. and Smith, A. E., A Genetic Approach to the Quadratic Assignment Problem. Computers and Operations Research, 1(1995), pp.855-865.
[86]. Thierens D (2006). Exploration and Exploitation Bias of Crossover and Path Relinking for Permutation Problems. Lecture Notes in Computer Science.4193:1028-1037
[87]. Tseng L Y and Liang S C (2006). A hybrid metaheuristic for the quadratic assignment problem. Computational Optimization and Applications. 34(1):85-113.
[88]. Voß S, Martello S , Osman I H and Roucairol C (1999). Meta-heuristics Advances and Trends in Local Search Paradigms for Optimization. Kluwer Academic Publishers: Dordrecht, The Netherlands
[89]. Vollmann T E and Buffa E S (1966).The Facilities Layout Problem in Perspective. Management Science. 12(10): 450-468
[90]. Xu J, Chiu S Y and Glover F (1998). Fine-tuning a tabu search algorithm with statistical tests. International Transactions in Operational Research 5 (3): 233–244.
[91]. Xu Y L, Lim M H, Ong Y S and Tang J (2006). A GA-ACO-Local Search Hybrid Algorithm for Solving Quadratic Assignment Problem, In: Proceedings of GECCO'06.
[92]. Yagiura M, Ibaraki T and Glover F (2006). A path relinking approach with ejection chains for the generalized assignment problem. European Journal of Operational Research. 169, 548–569.

[93]. Zhang G and Lai K (2006).Combining path relinking and genetic algorithms for the multiple-level warehouse layout problem. European Journal of Operational Research.169:413-425

[94]. Zitzler E and Thiele L (1998). Multiobjective optimization using evolutionary algorithms - A comparative case study.

[95]. Zitzler E and Thiele L (1999). Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto approach. IEEE Transactions on Evolutionary Computation, 3(4):257-271.

[96]. Zitzler E, Laumanns M, and Bleuler S (2004). A Tutorial on Evolutionary Multiobjective Optimization. In Xavier Gandibleux *et al.* (Eds). Metaheuristics for Multiobjective Optimization, Springer. Lecture Notes in Economics and Mathematical Systems. 535:3–37

APPENDICES

APPENDIX A: Test Cases of the Initial Comparison

File number	Problem size	Authors	Algorithm	Remarks
Bur26a	**26**	**R.E. Burkard and J. Offermann**	(OPT)	Asymmetric
Bur26b	**26**		(GRASP)	
Bur26c	**26**		(GRASP)	
Bur26d	**26**		(GRASP)	
Bur26e	**26**		(GRASP)	
Bur26f	**26**		(GRASP)	
Bur26g	**26**		(GRASP)	
Bur26h	**26**		(GRASP)	
Bur26	**26**		(GRASP)	
Chr12a	**12**	**N. Christofides and E. Benavent**	(OPT)	One matrix is the adjacency matrix of a weighted tree the other that of a complete graph
Chr12b	**12**		(OPT)	
Chr12c	**12**		(OPT)	
Chr15a	**15**		(OPT)	
Chr15b	**15**		(OPT)	
Chr15c	**15**		(OPT)	
Chr18a	**18**		(OPT)	
Chr18b	**18**		(OPT)	
Chr20a	**20**		(OPT)	
Chr20b	**20**		(OPT)	
Chr20c	**20**		(OPT)	
Chr22a	**22**		(OPT)	
Chr22b	**22**		(OPT)	
Chr22c	**25**		(OPT)	
Els19	**19**	**A.N. Elshafei**	(OPT)	hospital and the flow of patients between those locations
Esc16a	**16**	**B. Eschermann and H.J. Wunderlich**	(OPT)	application in computer science, from the testing of self-testable sequential circuits
Esc16b	**16**		(OPT)	
Esc16c	**16**		(OPT)	
Esc16d	**16**		(OPT)	
Esc16e	**16**		(OPT)	
Esc16f	**16**		(OPT)	
Esc16g	**16**		(OPT)	

Esc16h	**16**		(OPT)	
Esc16i	**16**		(OPT)	
Esc16j	**16**		(OPT)	
Esc32a	**32**		(Ro-TS)	
Esc32b	**32**		(Ro-TS)	
Esc32c	**32**		(SIM-1)	
Esc32d	**32**		(Ro-TS)	
Esc32e	**32**		(OPT)	
Esc32f	**32**		(OPT)	
Esc32g	**32**		(OPT)	
Esc64a	**64**		(SIM-1)	
Esc128	**128**		(GRASP)	
Had12	**12**	**S.W. Hadley, F. Rendl and H. Wolkowicz**	(OPT)	Manhattan distances of a connected cellular complex in the plane and flow matrix are drawn uniformly from the interval [1,n]
Had14	**14**		(OPT)	
Had16	**16**		(OPT)	
Had18	**18**		(OPT)	
Had20	**20**		(OPT)	
Kra30a	**30**	**J. Krarup and P.M. Pruzan**	(OPT)	The instances contain real world data and were used to plan the Klinikum Regensburg in Germany.
Kra30b	**30**	**J. Krarup and P.M. Pruzan**	(OPT)	The instances contain real world data and were used to plan the Klinikum Regensburg in Germany.
Kra32	**32**	**J. Krarup and P.M. Pruzan**	(OPT)	
Lipa20a	**20**	**Y. Li and P.M. Pardalos**	(OPT)	The generators provide asymmetric instances with known optimal solutions.
Lipa20b	**20**		(OPT)	
Lipa30a	**30**		(OPT)	
Lipa30b	**30**		(OPT)	
Lipa40a	**40**		(OPT)	
Lipa40b	**40**		(OPT)	
Lipa50a	**50**		(OPT)	
Lipa50b	**50**		(OPT)	
Lipa60a	**60**		(OPT)	
Lipa60b	**60**		(OPT)	
Lipa70a	**70**		(OPT)	
Lipa70b	**70**		(OPT)	
Lipa80a	**80**		(OPT)	

Lipa80b	**80**		(OPT)	
Lipa90a	**90**		(OPT)	
Lipa90b	**90**		(OPT)	
Nug12	**12**	**C.E. Nugent, T.E. Vollmann and J. Ruml**	(OPT)	The distance matrix contains Manhattan distances of rectangular grids.
Nug14	**14**		(OPT)	
Nug15	**15**		(OPT)	
Nug16a	**16**		(OPT)	
Nug16b	**16**		(OPT)	
Nug17	**17**		(OPT)	
Nug18	**18**		(OPT)	
Nug20	**20**		(OPT)	
Nug21	**21**		(OPT)	
Nug22	**22**		(OPT)	
Nug24	**24**		(OPT)	
Nug25	**25**		(OPT)	
Nug27	**27**		(OPT)	
Nug30	**30**		(OPT)	
Rou12	**12**	**C. Roucairol**	(OPT)	The entries of the matrices are chosen from the interval [1,100].
Rou15	**15**		(OPT)	
Rou20	**20**		(OPT)	
Scr12	**12**	**M. Scriabin and R.C. Vergin**	(OPT)	The distances of these problems are rectangular.
Scr15	**15**		(OPT)	
Scr20	**20**		(OPT)	The distances of these problems are rectangular.Taixxa are uniformly generated
Sko42	**42**	**J. Skorin-Kapov**	Ro-TS)	The distances of these problems are rectangular and the entries in flow matrices are pseudorandom numbers.
Sko49	**49**		Ro-TS)	
Sko56	**56**		Ro-TS)	
Sko64	**64**		Ro-TS)	
Sko72	**72**		Ro-TS)	
Sko81	**81**		(GEN)	
Sko90	**90**		Ro-TS)	
Sko100a	**100**		(GEN)	
Sko100b	**100**		(GEN)	
Sko100c	**100**		(GEN)	
Sko100d	**100**		(GEN)	
Sko100e	**100**		(GEN)	
Sko100f	**100**		(GEN)	
Ste36a	**36**	**L. Steinberg**	(OPT)	The three instances model the backboard wiring problem. The distances in the first one

				are Manhattan,
Ste36b	**36**		(OPT)	The three instances model the backboard wiring problem. The distances in the second squared Euclidean
Ste36c	**36**		(OPT)	The three instances model the backboard wiring problem. The distances in the third one Euclidean distances (multiplied by 1000).
Tai12a	**12**	**E.D. Taillard**	(OPT)	Tai*xx*a are uniformly generated
Tai12b	**12**		(OPT)	Tai*xx*b are asymmetric and randomly generated
Tai15a	**15**		(OPT)	Tai*xx*a are uniformly generated
Tai15b	**15**		(OPT)	Tai*xx*b are asymmetric and randomly generated
Tai17	**17**		(OPT)	Tai*xx*a are uniformly generated
Tai20a	**20**		(OPT)	Tai*xx*a are uniformly generated
Tai20b	**20**		(OPT)	Tai*xx*b are asymmetric and randomly generated
Tai25a	**25**		(Ro-TS)	Tai*xx*a are uniformly generated
Tai25b	**25**		(OPT)	Tai*xx*b are asymmetric and randomly generated
Tai30a	**30**		(Ro-TS)	Tai*xx*a are uniformly generated
Tai30b	**30**		(Ro-TS)	Tai*xx*b are asymmetric and randomly generated
Tai35a	**35**		(Ro-TS)	Tai*xx*a are uniformly generated
Tai35b	**35**		(Ro-TS)	Tai*xx*b are asymmetric and randomly generated
Tai40a	**40**		(Re-TS)	Tai*xx*a are uniformly generated
Tai40b	**40**		(Ro-TS)	Tai*xx*b are asymmetric and randomly generated
Tai50a	**50**		(GEN)	Tai*xx*a are uniformly generated
Tai50b	**50**		(Ro-TS)	Tai*xx*b are asymmetric and randomly generated
Tai60a	**60**		(Ro-TS)	Tai*xx*a are uniformly generated
Tai60b	**60**		(Ro-TS)	Tai*xx*b are asymmetric and randomly generated
Tai64	**64**		(Ro-TS)	Tai*xx*c occur in the generation of grey patterns
Tai80a	**80**		(Ro-TS)	Tai*xx*a are uniformly generated

Tai80b	**80**		(Ro-TS)	Tai*xx*b are asymmetric and randomly generated
Tai100b	**100**		(Re-TS)	Tai*xx*a are uniformly generated
Tai100a	**100**		(Ro-TS)	Tai*xx*b are asymmetric and randomly generated
Tho30	**30**	**U.W. Thonemann and A. Bölte**	(OPT)	The distances of these instances are rectangular.
Tho40	**40**		(SIM-2)	The distances of these instances are rectangular.
Tho50	**50**	**M.R. Wilhelm and T.L. Ward**	(SIM-2)	The distances of these problems are rectangular.

OPT: Optimally Solved
GRASP: Greedy Randomized Adaptive Search Procedure
Ro-TS: Robust Tabu Search
SIM-1: Simulated Annealing I
GEN: Genetic Hybrids
SIM-2: Simulated Annealing II

APPENDIX B: Average Losses of Random Keys-based Metaheuristics (After Parameter Tuning)

Name	Average Loss (L) of Algorithms L=(Value obtained –Best Known)/Best Known			
	GA-RK	IA-RK	SA-RK	TS-RK
Bur26a	0.0192	0.0184	0.0146	0.0020
Bur26b	0.0189	0.0192	0.0147	0.0032
Bur26c	0.0208	0.0213	0.0163	0.0020
Bur26d	0.0212	0.0210	0.0161	0.0025
Bur26e	0.0217	0.0211	0.0161	0.0016
Bur26f	0.0188	0.0192	0.0147	0.0026
Bur26g	0.0243	0.0236	0.0176	0.0031
Bur26h	0.0196	0.0197	0.0152	0.0019
Bur26	0.0000	0.0000	0.0000	0.0000
Chr12a	0.2917	0.2139	0.0849	0.1739
Chr12b	0.2252	0.1892	0.0611	0.3724
Chr12c	0.1550	0.1472	0.0413	0.1047
Chr15a	0.5907	0.6261	0.3701	0.1752
Chr15b	0.9604	0.8934	0.5690	0.3337
Chr15c	0.7993	0.7590	0.4875	0.2933
Chr18a	1.3320	1.2990	0.9348	0.2828
Chr18b	0.3349	0.3498	0.2263	0.0195
Chr20a	1.0384	1.0639	0.8098	0.1974
Chr20b	0.9651	0.9268	0.6938	0.1284
Chr20c	1.6919	1.6656	1.2430	0.4531
Chr22a	0.3082	0.2937	0.2446	0.0553
Chr22b	0.3058	0.3026	0.2389	0.0564
Chr22c	1.6234	1.6132	1.3442	0.2679
Els19	0.2231	0.2403	0.1485	0.2838
Esc16a	0.0165	0.0165	0.0006	0.0000
Esc16b	0.0000	0.0000	0.0000	0.0000
Esc16c	0.0235	0.0178	0.0033	0.0000
Esc16d	0.0625	0.0875	0.0000	0.0000
Esc16e	0.0200	0.0157	0.0000	0.0014
Esc16f	0.0000	0.0000	0.0000	0.0100
Esc16g	0.0169	0.0077	0.0000	0.0000
Esc16h	0.0000	0.0000	0.0000	0.0000
Esc16i	0.0000	0.0000	0.0000	0.0000
Esc16j	0.0000	0.0000	0.0000	0.0000
Esc32a	0.9714	0.9606	0.8382	0.0809
Esc32b	0.9105	0.9319	0.8510	0.1238
Esc32c	0.0630	0.0950	0.0665	0.0000
Esc32d	0.2470	0.2604	0.2094	0.0242
Esc32e	0.0000	0.0000	0.0000	0.0000
Esc32f	0.0000	0.0000	0.0000	0.0000
Esc32g	0.1787	0.1840	0.1462	0.0276

Esc64a	0.4231	0.4228	0.3576	0.0000
Esc128	1.8419	2.1531	2.0281	0.0250
Had12	0.0140	0.0133	0.0035	0.0025
Had14	0.0183	0.0190	0.0097	0.0038
Had16	0.0257	0.0254	0.0170	0.0022
Had18	0.0280	0.0273	0.0205	0.0047
Had20	0.0340	0.0354	0.0272	0.0053
Kra30a	0.2569	0.2589	0.2272	0.0435
Kra30b	0.2398	0.2442	0.2157	0.0273
Kra32	0.2674	0.2759	0.2510	0.0359
Lipa20a	0.0386	0.0386	0.0343	0.0194
Lipa20b	0.2004	0.1993	0.1846	0.0590
Lipa30a	0.0322	0.0322	0.0304	0.0154
Lipa30b	0.2252	0.2240	0.2163	0.1351
Lipa40a	0.0266	0.0265	0.0253	0.0119
Lipa40b	0.2511	0.2502	0.2443	0.1517
Lipa50a	0.0237	0.0236	0.0228	0.0105
Lipa50b	0.2484	0.2485	0.2438	0.1709
Lipa60a	0.0211	0.0210	0.0203	0.0091
Lipa60b	0.2644	0.2635	0.2600	0.1812
Lipa70a	0.0188	0.0187	0.0183	0.0080
Lipa70b	0.2707	0.2709	0.2682	0.1970
Lipa80a	0.0171	0.0171	0.0167	0.0071
Lipa80b	0.2807	0.2810	0.2778	0.2065
Lipa90a	0.0157	0.0157	0.0154	0.0066
Lipa90b	0.2817	0.2813	0.2794	0.2105
Nug12	0.0379	0.0386	0.0140	0.0156
Nug14	0.0685	0.0630	0.0423	0.0184
Nug15	0.0817	0.0811	0.0536	0.0104
Nug16a	0.0863	0.0844	0.0606	0.0188
Nug16b	0.0923	0.1028	0.0761	0.0180
Nug17	0.0940	0.0955	0.0670	0.0138
Nug18	0.1024	0.1019	0.0796	0.0158
Nug20	0.1161	0.1139	0.0907	0.0187
Nug21	0.1382	0.1378	0.1107	0.0174
Nug22	0.1398	0.1369	0.1118	0.0193
Nug24	0.1552	0.1550	0.1325	0.0196
Nug25	0.1448	0.1472	0.1259	0.0107
Nug27	0.1587	0.1538	0.1338	0.0197
Nug30	0.1623	0.1641	0.1445	0.0165
Rou12	0.0523	0.0473	0.0183	0.0203
Rou15	0.0974	0.0916	0.0689	0.0272

Rou20	0.1045	0.1027	0.0848	0.0155
Scr12	0.0526	0.0531	0.0159	0.0297
Scr15	0.1590	0.1719	0.1218	0.0601
Scr20	0.2961	0.3018	0.2316	0.0328
Sko42	0.1588	0.1622	0.1495	0.0166
Sko49	0.1517	0.1521	0.1433	0.0170
Sko56	0.1542	0.1563	0.1477	0.0172
Sko64	0.1458	0.1462	0.1379	0.0164
Sko72	0.1432	0.1434	0.1382	0.0163
Sko81	0.1371	0.1378	0.1338	0.0148
Sko90	0.1369	0.1371	0.1336	0.0156
Sko100a	0.1318	0.1313	0.1277	0.0136
Sko100b	0.1302	0.1311	0.1273	0.0146
Sko100c	0.1369	0.1371	0.1331	0.0158
Sko100d	0.1299	0.1325	0.1283	0.0148
Sko100e	0.1373	0.1390	0.1343	0.0159
Sko100f	0.1295	0.1298	0.1262	0.0142
Ste36a	0.5883	0.6019	0.5391	0.0472
Ste36b	1.5078	1.5468	1.3096	0.1122
Ste36c	0.5281	0.5285	0.4705	0.0511
Tai12a	0.0824	0.0741	0.0425	0.0351
Tai12b	0.0437	0.0325	0.0093	0.0960
Tai15a	0.0703	0.0669	0.0480	0.0183
Tai15b	0.0103	0.0095	0.0061	0.0030
Tai17	0.0954	0.0928	0.0736	0.0250
Tai20a	0.1136	0.1149	0.0983	0.0256
Tai20b	0.0905	0.0908	0.0721	0.1414
Tai25a	0.1206	0.1222	0.1072	0.0265
Tai25b	0.2643	0.2503	0.1784	0.1618
Tai30a	0.1209	0.1197	0.1089	0.0280
Tai30b	0.2267	0.2351	0.1750	0.0988
Tai35a	0.1288	0.1303	0.1212	0.0265
Tai35b	0.2716	0.2612	0.2151	0.0700
Tai40a	0.1328	0.1324	0.1253	0.0291
Tai40b	0.3487	0.3455	0.2989	0.0821
Tai50a	0.1365	0.1369	0.1300	0.0319
Tai50b	0.3380	0.3337	0.3019	0.0553
Tai60a	0.1343	0.1340	0.1285	0.0315
Tai60b	0.3567	0.3589	0.3337	0.0630
Tai64	0.0525	0.0508	0.0270	0.0049
Tai80a	0.1189	0.1185	0.1157	0.0257
Tai80b	0.3374	0.3421	0.3244	0.0420

Tai100b	0.1125	0.1128	0.1104	0.0236
Tai100a	0.3282	0.3286	0.3155	0.0429
Tho30	0.1951	0.1983	0.1721	0.0224
Tho40	0.2326	0.2354	0.2151	0.0208
Tho50	0.0859	0.0850	0.0798	0.0100

APPENDIX C: Average Losses of Permutation-based Metaheuristics (After Parameter Tuning)

Name	Average Loss (L) of Algorithms L=(Value obtained –Best Known)/Best Known			
	GA-PM	IA-PM	SA-PM	TS-PM
Bur26a	0.0090	0.0003	0.0146	0.0016
Bur26b	0.0089	0.0001	0.0147	0.0020
Bur26c	0.0101	0.0001	0.0163	0.0018
Bur26d	0.0089	0.0000	0.0161	0.0012
Bur26e	0.0089	0.0001	0.0161	0.0010
Bur26f	0.0092	0.0001	0.0147	0.0024
Bur26g	0.0096	0.0000	0.0176	0.0022
Bur26h	0.0090	0.0001	0.0152	0.0018
Bur26	0.0000	0.0000	0.0000	0.0000
Chr12a	0.1329	0.0000	0.0849	0.0963
Chr12b	0.0862	0.0000	0.0611	0.2590
Chr12c	0.1234	0.0014	0.0413	0.0595
Chr15a	0.4632	0.0126	0.3701	0.0802
Chr15b	0.5568	0.0095	0.5690	0.2140
Chr15c	0.5964	0.0849	0.4875	0.2124
Chr18a	0.9952	0.0764	0.9348	0.1991
Chr18b	0.2094	0.0005	0.2263	0.0096
Chr20a	0.8325	0.1092	0.8098	0.1216
Chr20b	0.8137	0.1058	0.6938	0.0999
Chr20c	1.1091	0.0550	1.2430	0.2842
Chr22a	0.2332	0.0383	0.2446	0.0538
Chr22b	0.2367	0.0442	0.2466	0.0434
Chr22c	1.2522	0.2159	1.3442	0.2286
Els19	0.2622	0.0009	0.1485	0.2740
Esc16a	0.0024	0.0000	0.0006	0.0000
Esc16b	0.0000	0.0000	0.0000	0.0000
Esc16c	0.0080	0.0000	0.0033	0.0000
Esc16d	0.0200	0.0000	0.0000	0.0000
Esc16e	0.0043	0.0000	0.0000	0.0000
Esc16f	0.0000	0.0000	0.0000	0.0000
Esc16g	0.0215	0.0000	0.0000	0.0000
Esc16h	0.0000	0.0000	0.0000	0.0000
Esc16i	0.0000	0.0000	0.0000	0.0000
Esc16j	0.0000	0.0000	0.0000	0.0000
Esc32a	0.6455	0.1062	0.8382	0.0452
Esc32b	0.8929	0.1281	0.8510	0.0748
Esc32c	0.0074	0.0000	0.0665	0.0000
Esc32d	0.1106	0.0002	0.2094	0.0126
Esc32e	0.0000	0.0000	0.0000	0.0000

Esc32f	0.0000	0.0000	0.0000	0.0000
Esc32g	0.0819	0.0011	0.1462	0.0059
Esc64a	0.1045	0.0000	0.3576	0.0000
Esc128	1.5388	0.3631	2.0281	0.0163
Had12	0.0072	0.0000	0.0035	0.0043
Had14	0.0129	0.0000	0.0097	0.0026
Had16	0.0167	0.0000	0.0170	0.0003
Had18	0.0200	0.0000	0.0205	0.0021
Had20	0.0214	0.0000	0.0272	0.0033
Kra30a	0.1741	0.0313	0.2272	0.0358
Kra30b	0.1458	0.0188	0.2157	0.0189
Kra32	0.1682	0.0326	0.2510	0.0335
Lipa20a	0.0343	0.0011	0.0343	0.0160
Lipa20b	0.0000	0.0000	0.1846	0.0292
Lipa30a	0.0286	0.0157	0.0304	0.0128
Lipa30b	0.0000	0.0325	0.2163	0.0883
Lipa40a	0.0233	0.0142	0.0253	0.0123
Lipa40b	0.0000	0.1583	0.2443	0.1287
Lipa50a	0.0207	0.0132	0.0228	0.0111
Lipa50b	0.0000	0.1891	0.2438	0.1530
Lipa60a	0.0183	0.0119	0.0203	0.0098
Lipa60b	0.0000	0.2082	0.2600	0.1903
Lipa70a	0.0164	0.0109	0.0183	0.0087
Lipa70b	0.0000	0.2183	0.2682	0.1951
Lipa80a	0.0150	0.0100	0.0167	0.0080
Lipa80b	0.0000	0.2302	0.2778	0.2095
Lipa90a	0.0138	0.0094	0.0154	0.0075
Lipa90b	0.0000	0.2337	0.2794	0.2178
Nug12	0.0302	0.0000	0.0140	0.0100
Nug14	0.0500	0.0002	0.0423	0.0141
Nug15	0.0560	0.0000	0.0536	0.0040
Nug16a	0.0584	0.0005	0.0606	0.0100
Nug16b	0.0716	0.0000	0.0761	0.0064
Nug17	0.0639	0.0008	0.0670	0.0067
Nug18	0.0717	0.0023	0.0796	0.0113
Nug20	0.0831	0.0012	0.0907	0.0092
Nug21	0.0900	0.0017	0.1107	0.0104
Nug22	0.0846	0.0012	0.1118	0.0096
Nug24	0.1111	0.0070	0.1325	0.0113
Nug25	0.1029	0.0049	0.1259	0.0052
Nug27	0.1071	0.0094	0.1338	0.0163
Nug30	0.1192	0.0159	0.1445	0.0125
Rou12	0.0351	0.0000	0.0183	0.0142

Rou15	0.0747	0.0003	0.0689	0.0197
Rou20	0.0822	0.0055	0.0848	0.0108
Scr12	0.0370	0.0000	0.0159	0.0273
Scr15	0.1143	0.0000	0.1218	0.0337
Scr20	0.2158	0.0054	0.2316	0.0161
Sko42	0.1251	0.0265	0.1495	0.0157
Sko49	0.1125	0.0280	0.1433	0.0168
Sko56	0.1200	0.0362	0.1477	0.0156
Sko64	0.1123	0.0381	0.1379	0.0161
Sko72	0.1101	0.0401	0.1382	0.0157
Sko81	0.1048	0.0407	0.1338	0.0127
Sko90	0.1061	0.0429	0.1336	0.0137
Sko100a	0.1047	0.0438	0.1277	0.0126
Sko100b	0.1007	0.0423	0.1273	0.0135
Sko100c	0.1069	0.0460	0.1331	0.0159
Sko100d	0.1005	0.0430	0.1283	0.0143
Sko100e	0.1053	0.0464	0.1343	0.0156
Sko100f	0.0998	0.0426	0.1262	0.0367
Ste36a	0.5410	0.0570	0.5391	0.0446
Ste36b	0.8997	0.0822	1.3096	0.0927
Ste36c	0.3864	0.0403	0.4705	0.0387
Tai12a	0.0579	0.0000	0.0425	0.0241
Tai12b	0.0269	0.0000	0.0093	0.0743
Tai15a	0.0539	0.0006	0.0480	0.0092
Tai15b	0.0064	0.0000	0.0061	0.0024
Tai17	0.0744	0.0067	0.0736	0.0172
Tai20a	0.0953	0.0143	0.0983	0.0197
Tai20b	0.0554	0.0006	0.0721	0.1472
Tai25a	0.0998	0.0276	0.1072	0.0231
Tai25b	0.1512	0.0027	0.1784	0.0912
Tai30a	0.0979	0.0315	0.1089	0.0229
Tai30b	0.1378	0.0041	0.1750	0.1246
Tai35a	0.1067	0.0390	0.1212	0.0276
Tai35b	0.1300	0.0098	0.2151	0.0656
Tai40a	0.1112	0.0458	0.1253	0.0312
Tai40b	0.2305	0.0160	0.2989	0.0657
Tai50a	0.1146	0.0531	0.1300	0.0359
Tai50b	0.2080	0.0257	0.3019	0.0420
Tai60a	0.1127	0.0569	0.1285	0.0376
Tai60b	0.2149	0.0370	0.3337	0.0420
Tai64	0.0183	0.0011	0.0270	0.0060
Tai80a	0.1001	0.0544	0.1157	0.0332

Tai80b	0.2291	0.0695	0.3244	0.0349
Tai100b	0.0948	0.0552	0.1104	0.0352
Tai100a	0.2027	0.0759	0.3155	0.0312
Tho30	0.1397	0.0144	0.1721	0.0202
Tho40	0.1733	0.0347	0.2151	0.0228
Tho50	0.0638	0.0147	0.0798	0.0086

APPENDIX D: Problem Characteristics of the 130 QAPLIB Instances

Name	Problem size	Flow Dominance	Distance Dominance	Sparsity Flow Dominance	Sparsity Distance Dominance	Sparsity
Bur26a	26	274.947	15.0854	0.0000000	0.223373	0.223373
Bur26b	26	274.947	15.9127	0.0000000	0.223373	0.223373
Bur26c	26	228.396	15.0854	0.0000000	0.257396	0.257396
Bur26d	26	228.396	15.9127	0.0000000	0.257396	0.257396
Bur26e	26	253.995	15.0854	0.0000000	0.312130	0.312130
Bur26f	26	274.947	15.9127	0.0000000	0.223373	0.223373
Bur26g	26	253.995	15.9127	0.0000000	0.312130	0.312130
Bur26h	26	279.894	15.0854	0.0000000	0.211538	0.211538
Bur26	26	60.000	60.000	0.0000000	0.0000000	0.0000000
Chr12a	12	63.427	309.055	0.847222	0.0972222	0.7499998
Chr12b	12	63.427	309.055	0.847222	0.0972222	0.7499998
Chr12c	12	63.427	309.055	0.847222	0.0972222	0.7499998
Chr15a	15	69.8909	327.68	0.875556	0.0666667	0.8088893
Chr15b	15	69.8909	327.68	0.875556	0.0666667	0.8088893
Chr15c	15	69.8909	327.68	0.875556	0.0666667	0.8088893
Chr18a	18	63.196	351.138	0.895062	0.0555556	0.8395064
Chr18b	18	56.9507	356.87	0.895062	0.0555556	0.8395064
Chr20a	20	59.4589	346.373	0.9050000	0.0500000	0.8550000
Chr20b	20	59.4589	346.373	0.9050000	0.0500000	0.8550000
Chr20c	20	65.7126	346.373	0.9050000	0.0500000	0.8550000
Chr22a	22	66.9564	421.056	0.913223	0.0454545	0.8677685
Chr22b	22	66.9564	421.056	0.913223	0.0454545	0.8677685
Chr22c	25	57.9713	424.268	0.923200	0.0400000	0.8832000
Els19	19	531.017	52.1017	0.0526316	0.689751	0.6371194
Esc16a	16	84.7981	170.383	0.7031250	0.312500	0.3906250
Esc16b	16	84.7981	75.6532	0.28125	0.312500	0.0312500
Esc16c	16	84.7981	133.361	0.601563	0.312500	0.289063
Esc16d	16	84.7981	235.686	0.835938	0.312500	0.523438
Esc16e	16	84.7981	249.464	0.835938	0.312500	0.523438
Esc16f	16	84.7981	249.464	1.000000	0.312500	0.687500
Esc16g	16	84.7981	254.62	0.835938	0.312500	0.523438
Esc16h	16	84.7981	151.184	0.101563	0.312500	0.210937
Esc16i	16	84.7981	296.854	0.882813	0.312500	0.570313
Esc16j	16	84.7981	322.515	0.90625	0.312500	0.593750
Esc32a	32	69.2714	281.567	0.855469	0.312500	0.667969
Esc32b	32	69.2714	208.268	0.789063	0.312500	0.601563

Esc32c	32	69.2714	200.266	0.744141	0.187500	0.556641
Esc32d	32	69.2714	235.484	0.824219	0.187500	0.636719
Esc32e	32	69.2714	1091.4	0.988281	0.187500	0.800781
Esc32f	32	69.2714	849.785	0.982422	0.187500	0.794922
Esc32g	32	69.2714	187.852	0.724609	0.187500	0.537109
Esc64a	64	59.1612	571.567	0.968262	0.109375	0.858887
Esc128	128	52.0396	1153.98	0.992432	0.0625000	0.929932
Had12	12	50.8555	63.3506	0.0833333	0.0833333	0.0000000
Had14	14	49.5829	66.7922	0.0714286	0.0714286	0.0000000
Had16	16	48.4975	64.9555	0.0625000	0.0625000	0.0000000
Had18	18	47.2052	63.7798	0.0555556	0.0555556	0.0000000
Had20	20	46.0146	64.3235	0.0500000	0.0500000	0.0000000
Kra30a	30	149.98	49.2257	0.0333333	0.633333	0.5999997
Kra30b	30	149.98	49.9925	0.0333333	0.633333	0.5999997
Kra32	32	49.0145	164.222	0.677734	0.031250	0.646484
Lipa20a	20	45.6447	39.7857	0.0975000	0.050000	0.047500
Lipa20b	20	45.6447	68.9274	0.1075000	0.050000	0.057500
Lipa30a	30	42.9374	32.1813	0.0655556	0.0333333	0.0322223
Lipa30b	30	42.9374	64.0566	0.0677778	0.0333333	0.0344445
Lipa40a	40	42.0834	27.7437	0.049375	0.0250000	0.024375
Lipa40b	40	42.0834	64.1591	0.051875	0.0250000	0.026875
Lipa50a	50	41.1717	24.7485	0.039600	0.0200000	0.019600
Lipa50b	50	41.1717	61.7666	0.041200	0.0250000	0.021200
Lipa60a	60	42.0981	22.5525	0.0330556	0.0166667	0.0163889
Lipa60b	60	42.0981	60.457	0.0311111	0.0166667	0.0144444
Lipa70a	70	41.9168	20.8536	0.0283673	0.0142857	0.0140816
Lipa70b	70	41.9168	60.3477	0.0302041	0.0142857	0.0159184
Lipa80a	80	42.2546	19.4886	0.0248438	0.0125000	0.0123438
Lipa80b	80	42.2546	60.1628	0.0257812	0.0125000	0.0132812
Lipa90a	90	41.8468	18.3608	0.0220988	0.0111111	0.0109877
Lipa90b	90	41.8468	60.0246	0.0216049	0.0111111	0.0104938
Nug12	12	116.987	57.0900	0.0833333	0.3750000	0.2916667
Nug14	14	103.832	56.8944	0.0714286	0.306122	0.2346934
Nug15	15	106.713	56.7082	0.0666667	0.333333	0.2666663
Nug16a	16	100.935	57.4464	0.0625000	0.273438	0.2109380
Nug16b	16	115.822	54.8795	0.0625000	0.34375	0.2812500
Nug17	17	105.009	56.3567	0.0588235	0.301038	0.2422145
Nug18	18	104.372	55.0199	0.0555556	0.302469	0.2469134
Nug20	20	103.775	54.17	0.0500000	0.295000	0.245000
Nug21	21	117.194	57.4498	0.047619	0.378685	0.331066
Nug22	22	114.334	64.1521	0.0454545	0.367769	0.3223145
Nug24	24	112.881	54.177	0.0416667	0.357639	0.3159723
Nug25	25	110.851	53.0755	0.0400000	0.3600000	0.3200000
Nug27	27	58.6547	111.479	0.360768	0.037037	0.323731
Nug30	30	112.479	52.7545	0.0333333	0.348889	0.3155557
Rou12	12	71.7881	67.2871	0.0833333	0.0972222	0.0138889
Rou15	15	69.227	68.8924	0.0666667	0.0755556	0.0088889

Rou20	20	64.4326	65.6508	0.055000	0.060000	0.0050000
Scr12	12	57.09	257.382	0.611111	0.0833333	0.5277777
Scr15	15	55.0436	248.302	0.626667	0.0666667	0.5600003
Scr20	20	54.17	254.652	0.690000	0.0500000	0.6400000
Sko42	42	108.483	51.9576	0.0238095	0.316327	0.2925175
Sko49	49	109.379	51.5496	0.0204082	0.324448	0.3040398
Sko56	56	110.53	51.4637	0.0178571	0.323342	0.3054849
Sko64	64	108.377	51.1829	0.015625	0.323242	0.307617
Sko72	72	107.128	51.1362	0.0138889	0.312886	0.2989971
Sko81	81	106.606	50.9328	0.0123457	0.306813	0.2944673
Sko90	90	107.528	50.9077	0.0111111	0.315802	0.3046909
Sko100a	100	106.641	50.7545	0.0100000	0.313800	0.3038000
Sko100b	100	108.368	50.7545	0.0100000	0.317200	0.3072000
Sko100c	100	108.072	50.7545	0.0100000	0.325600	0.3156000
Sko100d	100	109.191	50.7545	0.0100000	0.326600	0.3166000
Sko100e	100	110.399	50.7545	0.0100000	0.326800	0.3168000
Sko100f	100	108.337	50.7545	0.0100000	0.324600	0.3146000
Ste36a	36	400.305	55.6458	0.0277778	0.734568	0.7067902
Ste36b	36	400.305	100.79	0.0277778	0.734568	0.7067902
Ste36c	36	400.305	55.9034	0.0277778	0.734568	0.7067902
Tai12a	12	75.0259	69.5492	0.0833333	0.111111	0.0277777
Tai12b	12	300.652	79.4872	0.111111	0.493056	0.3819450
Tai15a	15	70.7203	63.9191	0.0755556	0.0844444	0.0088888
Tai15b	15	313.633	262.898	0.0666667	0.506667	0.4400003
Tai17	17	68.9928	64.2971	0.0795848	0.0726644	0.0069204
Tai20a	20	64.9035	67.0207	0.0650000	0.0650000	0.0000000
Tai20b	20	333.231	128.254	0.0500000	0.4600000	0.4100000
Tai25a	25	64.2969	61.8130	0.0432000	0.0560000	0.0128000
Tai25b	25	310.400	87.0155	0.0400000	0.4272000	0.3872000
Tai30a	30	63.2111	58.0009	0.0400000	0.0466667	0.0066667
Tai30b	30	323.909	85.1987	0.0355556	0.465556	0.4300004
Tai35a	35	61.5701	61.6406	0.0432653	0.0383673	0.004898
Tai35b	35	309.621	78.6582	0.0285714	0.5526530	0.5240816
Tai40a	40	60.2352	63.1039	0.0387500	0.0337500	0.0050000
Tai40b	40	317.219	66.746	0.0250000	0.5275000	0.5025000
Tai50a	50	62.2491	60.7488	0.0328000	0.0240000	0.0088000
Tai50b	50	313.914	73.4436	0.0200000	0.5676000	0.5476000
Tai60a	60	60.8573	61.4126	0.0238889	0.0272222	0.0033333
Tai60b	60	317.823	76.8319	0.0166667	0.5644440	0.5477773
Tai64	64	127.835	482.103	0.9587400	0.0156250	0.943115
Tai80a	80	60.3793	59.2238	0.0228125	0.0212500	0.0015625
Tai80b	80	323.174	64.046	0.0125000	0.5648440	0.552344
Tai100b	100	60.3121	59.338	0.0180000	0.0212000	0.0032000
Tai100a	100	321.342	80.4247	0.0100000	0.5615000	0.5515000
Tho30	30	137.863	59.2507	0.0333333	0.517778	0.4844447
Tho40	40	155.544	53.202	0.0250000	0.610000	0.5850000
Tho50	50	66.6628	54.1983	0.0200000	0.120800	0.1008000

APPENDIX E: Average Losses of Random Keys-based Metaheuristics (After POS_PR Augmentation)

Name	Average Loss (L) of Algorithms L=(Value obtained –Best Known)/Best Known			
	GA-RK-PR	IA-RK-PR	SA-RK-PR	TS-RK-PR
Bur26a	0.0056	0.0072	0.0173	0.0238
Bur26b	0.0054	0.0088	0.0188	0.0299
Bur26c	0.0053	0.0094	0.0207	0.0321
Bur26d	0.0035	0.0089	0.0209	0.0291
Bur26e	0.0054	0.0093	0.0209	0.0300
Bur26f	0.0051	0.0078	0.0176	0.0241
Bur26g	0.0057	0.0092	0.0228	0.0322
Bur26h	0.0057	0.0092	0.0189	0.0258
Bur26	0.0000	0.0000	0.0000	0.0000
Chr12a	0.1481	0.3122	0.1734	0.9827
Chr12b	0.1046	0.3030	0.1555	0.9639
Chr12c	0.1279	0.2678	0.1142	0.6302
Chr15a	0.3391	0.4961	0.5575	1.3633
Chr15b	0.5163	0.6409	0.8407	1.7759
Chr15c	0.4351	0.6365	0.6823	1.5300
Chr18a	0.6838	0.8007	1.1767	1.6429
Chr18b	0.1155	0.2122	0.3162	0.4780
Chr20a	0.5967	0.6602	0.9748	1.2647
Chr20b	0.3748	0.5708	0.8627	1.1689
Chr20c	0.7675	1.1017	1.5605	2.4316
Chr22a	0.1659	0.1947	0.2943	0.4064
Chr22b	0.1745	0.2053	0.2838	0.3630
Chr22c	0.9309	1.0391	1.5303	1.8231
Els19	0.0698	0.2525	0.2194	0.6961
Esc16a	0.0035	0.0294	0.0141	0.0500
Esc16b	0.0000	0.0000	0.0000	0.0010
Esc16c	0.0005	0.0098	0.0160	0.0718
Esc16d	0.0150	0.0750	0.0250	0.1975
Esc16e	0.0214	0.0714	0.0086	0.0957
Esc16f	0.0000	0.0000	0.0000	-1.0000
Esc16g	0.0000	0.0692	0.0092	0.1877

Esc16h	0.0000	0.0007	0.0000	0.0229
Esc16i	0.0000	0.0086	0.0000	0.1571
Esc16j	0.0000	0.0700	0.0000	0.2500
Esc32a	0.5040	0.4846	0.9262	0.9200
Esc32b	0.5376	0.6190	0.9114	0.8781
Esc32c	0.0009	0.0310	0.0914	0.1206
Esc32d	0.1012	0.1356	0.2494	0.2480
Esc32e	0.0000	0.0000	0.0000	1.2600
Esc32f	0.0000	0.1067	0.0000	0.5067
Esc32g	0.0574	0.0863	0.1779	0.1629
Esc64a	0.1579	0.2166	0.4376	0.4248
Esc128	1.3494	1.6113	2.2138	2.0294
Had12	0.0040	0.0089	0.0106	0.0381
Had14	0.0028	0.0071	0.0151	0.0441
Had16	0.0037	0.0112	0.0247	0.0402
Had18	0.0072	0.0148	0.0275	0.0358
Had20	0.0085	0.0145	0.0333	0.0502
Kra30a	0.1161	0.1504	0.2529	0.2457
Kra30b	0.0930	0.1339	0.2336	0.2480
Kra32	0.1416	0.1650	0.2733	0.2619
Lipa20a	0.0304	0.0318	0.0374	0.0413
Lipa20b	0.1661	0.1730	0.1945	0.2043
Lipa30a	0.0258	0.0256	0.0317	0.0320
Lipa30b	0.1959	0.1964	0.2230	0.2255
Lipa40a	0.0213	0.0212	0.0264	0.0258
Lipa40b	0.2243	0.2226	0.2491	0.2442
Lipa50a	0.0195	0.0191	0.0234	0.0225
Lipa50b	0.2264	0.2227	0.2479	0.2388
Lipa60a	0.0176	0.0171	0.0210	0.0196
Lipa60b	0.2440	0.2404	0.2627	0.2486
Lipa70a	0.0161	0.0154	0.0187	0.0224
Lipa70b	0.2536	0.2496	0.2709	0.2586
Lipa80a	0.0147	0.0142	0.0170	0.0212
Lipa80b	0.2650	0.2598	0.2803	0.2671
Lipa90a	0.0136	0.0132	0.0157	0.0196
Lipa90b	0.2671	0.2627	0.2814	0.2709
Nug12	0.0253	0.0461	0.0323	0.0947
Nug14	0.0361	0.0529	0.0558	0.0968
Nug15	0.0300	0.0496	0.0739	0.1364
Nug16a	0.0412	0.0557	0.0802	0.1207
Nug16b	0.0418	0.0677	0.0902	0.1357
Nug17	0.0384	0.0542	0.0855	0.1265

Nug18	0.0434	0.0625	0.0955	0.1343
Nug20	0.0447	0.0627	0.1071	0.1415
Nug21	0.0502	0.0728	0.1343	0.1592
Nug22	0.0319	0.0703	0.1315	0.1588
Nug24	0.0656	0.0909	0.1507	0.1647
Nug25	0.0575	0.0820	0.1441	0.1523
Nug27	0.0830	0.0882	0.1517	0.1573
Nug30	0.0698	0.0998	0.1564	0.1693
Rou12	0.0388	0.0524	0.0407	0.0926
Rou15	0.0626	0.0723	0.0872	0.1348
Rou20	0.0634	0.0702	0.0994	0.1147
Scr12	0.0422	0.0703	0.0455	0.1504
Scr15	0.0913	0.1184	0.1633	0.2452
Scr20	0.1297	0.1569	0.2730	0.3627
Sko42	0.0832	0.1052	0.1600	0.1568
Sko49	0.0831	0.1009	0.1518	0.1449
Sko56	0.0860	0.1055	0.1567	0.1417
Sko64	0.0874	0.1015	0.1441	0.1321
Sko72	0.0875	0.1033	0.1428	0.1357
Sko81	0.0832	0.1006	0.1385	0.1222
Sko90	0.0875	0.1012	0.1374	0.1306
Sko100a	0.0872	0.0989	0.1318	0.1185
Sko100b	0.0861	0.0977	0.1308	0.1192
Sko100c	0.0918	0.1057	0.1376	0.1242
Sko100d	0.0875	0.0994	0.1324	0.1173
Sko100e	0.0918	0.1051	0.1385	0.1288
Sko100f	0.0838	0.0983	0.1401	0.1292
Ste36a	0.2329	0.3313	0.6106	0.6105
Ste36b	0.5444	0.7251	1.4749	1.5574
Ste36c	0.1865	0.2843	0.5079	0.5186
Tai12a	0.0623	0.0860	0.0674	0.1354
Tai12b	0.0141	0.0792	0.0319	0.2208
Tai15a	0.0442	0.0564	0.0631	0.0921
Tai15b	0.0049	0.0060	0.0115	0.5088
Tai17	0.0657	0.0684	0.0908	0.1164
Tai20a	0.0699	0.0831	0.1103	0.1291
Tai20b	0.0325	0.1231	0.0890	0.3515
Tai25a	0.0826	0.0825	0.1207	0.1329
Tai25b	0.1000	0.1757	0.2395	0.5111
Tai30a	0.0842	0.0830	0.1189	0.1205
Tai30b	0.0835	0.1759	0.2373	0.3452
Tai35a	0.0934	0.0903	0.1307	0.1287

Tai35b	0.0810	0.1692	0.2550	0.3266
Tai40a	0.0993	0.0944	0.1351	0.1287
Tai40b	0.1276	0.1994	0.3306	0.3368
Tai50a	0.1076	0.1017	0.1382	0.1279
Tai50b	0.1212	0.1887	0.3373	0.3307
Tai60a	0.1059	0.1020	0.1350	0.1291
Tai60b	0.1302	0.2217	0.3454	0.3220
Tai64	0.0066	0.0700	0.0601	0.0877
Tai80a	0.0986	0.0926	0.1235	0.1155
Tai80b	0.1676	0.2282	0.3341	0.2883
Tai100b	0.0963	0.0919	0.1194	0.1047
Tai100a	0.1776	0.2315	0.3278	0.3074
Tho30	0.0694	0.1164	0.1933	0.2172
Tho40	0.1111	0.1406	0.2285	0.2196
Tho50	0.0443	0.0545	0.0848	0.0844

APPENDIX F: Average Losses of Permutation-based Metaheuristics (After POS_PR Augmentation)

Name	Average Loss (L) of Algorithms L=(Value obtained –Best Known)/Best Known			
	GA-PM-PR	IA-PM-PR	SA-PM-PR	TS-PM-PR
Bur26a	0.001	0.000	0.016	0.000
Bur26b	0.000	0.000	0.018	0.000
Bur26c	0.000	0.000	0.018	0.000
Bur26d	0.000	0.000	0.018	0.000
Bur26e	0.000	0.000	0.018	0.000
Bur26f	0.000	0.000	0.016	0.000
Bur26g	0.000	0.000	0.020	0.000
Bur26h	0.000	0.000	0.016	0.000
Bur26	0.000	0.000	0.000	0.000
Chr12a	0.002	0.000	0.219	0.000
Chr12b	0.000	0.000	0.262	0.000
Chr12c	0.005	0.002	0.133	0.000
Chr15a	0.023	0.013	0.410	0.000
Chr15b	0.043	0.014	0.747	0.000
Chr15c	0.069	0.072	0.568	0.000
Chr18a	0.138	0.091	1.108	0.036
Chr18b	0.001	0.000	0.251	0.000
Chr20a	0.141	0.107	0.930	0.002
Chr20b	0.132	0.113	0.740	0.054
Chr20c	0.108	0.058	1.301	0.047
Chr22a	0.052	0.037	0.298	0.016
Chr22b	0.048	0.045	0.260	0.026
Chr22c	0.270	0.231	1.450	0.097
Els19	0.002	0.001	0.186	0.000
Esc16a	0.000	0.000	0.024	0.000
Esc16b	0.000	0.000	0.000	0.000
Esc16c	0.000	0.000	0.017	0.000
Esc16d	0.000	0.000	0.050	0.000
Esc16e	0.000	0.000	0.001	0.000
Esc16f	0.000	0.000	0.000	0.000
Esc16g	0.000	0.000	0.014	0.000

Esc16h	0.000	0.000	0.010	0.000
Esc16i	0.000	0.000	0.040	0.000
Esc16j	0.000	0.000	0.015	0.000
Esc32a	0.128	0.101	0.805	0.000
Esc32b	0.172	0.131	0.771	0.000
Esc32c	0.000	0.000	0.074	0.000
Esc32d	0.000	0.001	0.199	0.000
Esc32e	0.000	0.000	0.280	0.000
Esc32f	0.000	0.000	0.233	0.000
Esc32g	0.004	0.001	0.142	0.000
Esc64a	0.000	0.000	0.358	0.000
Esc128	0.228	0.359	1.824	0.000
Had12	0.000	0.000	0.006	0.000
Had14	0.000	0.000	0.021	0.000
Had16	0.000	0.000	0.022	0.000
Had18	0.000	0.000	0.025	0.000
Had20	0.000	0.000	0.030	0.018
Kra30a	0.035	0.031	0.225	0.002
Kra30b	0.020	0.017	0.214	0.000
Kra32	0.030	0.033	0.248	0.000
Lipa20a	0.007	0.002	0.035	0.000
Lipa20b	0.000	0.000	0.189	0.015
Lipa30a	0.018	0.016	0.030	0.000
Lipa30b	0.027	0.033	0.216	0.011
Lipa40a	0.015	0.014	0.025	0.000
Lipa40b	0.125	0.165	0.243	0.010
Lipa50a	0.014	0.013	0.023	0.000
Lipa50b	0.185	0.189	0.244	0.008
Lipa60a	0.013	0.012	0.020	0.008
Lipa60b	0.212	0.208	0.258	0.008
Lipa70a	0.012	0.011	0.019	0.008
Lipa70b	0.222	0.219	0.267	0.007
Lipa80a	0.011	0.010	0.017	0.008
Lipa80b	0.235	0.230	0.278	0.006
Lipa90a	0.010	0.009	0.015	0.014
Lipa90b	0.238	0.234	0.277	0.000
Nug12	0.001	0.000	0.022	0.000
Nug14	0.001	0.000	0.054	0.000
Nug15	0.000	0.000	0.059	0.000
Nug16a	0.005	0.001	0.069	0.000
Nug16b	0.000	0.000	0.077	0.000
Nug17	0.004	0.001	0.073	0.000

Nug18	0.007	0.001	0.079	0.000
Nug20	0.006	0.002	0.096	0.000
Nug21	0.005	0.002	0.114	0.000
Nug22	0.001	0.001	0.125	0.000
Nug24	0.012	0.007	0.138	0.000
Nug25	0.008	0.004	0.129	0.000
Nug27	0.013	0.009	0.138	0.001
Nug30	0.021	0.016	0.154	0.000
Rou12	0.001	0.000	0.036	0.000
Rou15	0.005	0.000	0.078	0.002
Rou20	0.014	0.007	0.083	0.000
Scr12	0.000	0.000	0.046	0.000
Scr15	0.006	0.000	0.133	0.000
Scr20	0.013	0.005	0.233	0.008
Sko42	0.034	0.026	0.157	0.010
Sko49	0.035	0.030	0.142	0.008
Sko56	0.041	0.036	0.147	0.011
Sko64	0.045	0.036	0.139	0.010
Sko72	0.046	0.041	0.140	0.005
Sko81	0.046	0.041	0.131	0.008
Sko90	0.049	0.043	0.134	0.007
Sko100a	0.052	0.045	0.128	0.008
Sko100b	0.050	0.043	0.131	0.008
Sko100c	0.054	0.046	0.134	0.011
Sko100d	0.048	0.043	0.128	0.009
Sko100e	0.053	0.047	0.138	0.011
Sko100f	0.049	0.043	0.127	0.011
Ste36a	0.063	0.057	0.574	0.000
Ste36b	0.095	0.083	1.312	0.001
Ste36c	0.040	0.038	0.484	0.000
Tai12a	0.000	0.000	0.051	0.000
Tai12b	0.000	0.000	0.020	0.000
Tai15a	0.004	0.001	0.060	0.000
Tai15b	0.000	0.000	0.008	0.000
Tai17	0.015	0.006	0.081	0.007
Tai20a	0.022	0.014	0.098	0.000
Tai20b	0.001	0.001	0.080	0.009
Tai25a	0.033	0.026	0.103	0.001
Tai25b	0.002	0.003	0.208	0.015
Tai30a	0.038	0.031	0.107	0.003
Tai30b	0.008	0.004	0.208	0.015
Tai35a	0.045	0.038	0.122	0.005

Tai35b	0.009	0.010	0.230	0.020
Tai40a	0.053	0.045	0.125	0.011
Tai40b	0.034	0.013	0.287	0.022
Tai50a	0.060	0.053	0.126	0.003
Tai50b	0.042	0.026	0.312	0.027
Tai60a	0.064	0.057	0.130	0.004
Tai60b	0.050	0.037	0.345	0.000
Tai64	0.002	0.001	0.031	0.021
Tai80a	0.060	0.055	0.117	0.016
Tai80b	0.084	0.069	0.323	0.025
Tai100b	0.061	0.055	0.110	0.011
Tai100a	0.082	0.075	0.325	0.005
Tho30	0.021	0.014	0.170	0.006
Tho40	0.043	0.035	0.215	0.003
Tho50	0.018	0.015	0.081	0.003

APPENDIX G: Classification of QAPLIB Using POS_PR Augmentation and Problem Characteristics

Flow Dominance (FD): LOW (<=60), Moderate (60-200), and High (200>)
Distance Dominance (DD): LOW (<=100), Moderate (100-200), and High (200>)
Sparsity: LOW (<=0.5), Moderate (0.5-0.7), and High (0.7>)
Coefficient of Correlation: LOW (<=∓0.1), Moderate (∓0.1 - ∓ 0.5), and High (∓>0.5)
Problem Size: LOW (<=40), Moderate (40-75), and High (75>)

Name	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	Rank 8	FD	DD	Sparsity	Correlation N: Negative P: Positive	Problem Size
Bur26a	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**HIGH**	**LOW**	**LOW**	**LOW (N)**	**LOW**
Bur26b	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**HIGH**	**LOW**	**LOW**	**LOW (N)**	**LOW**
Bur26c	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**HIGH**	**LOW**	**LOW**	**LOW (N)**	**LOW**
Bur26d	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**HIGH**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Bur26e	TS-PM-PR	GA-PM-PR	IA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**HIGH**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Bur26f	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**HIGH**	**LOW**	**LOW**	**LOW (N)**	**LOW**
Bur26g	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**HIGH**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Bur26h	TS-PM-PR	GA-PM-PR	IA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**HIGH**	**LOW**	**LOW**	**LOW (N)**	**LOW**
Bur26	**ALL**	**ALL**	**ALL**	**ALL**	**ALL**	**ALL**	**ALL**	**ALL**	**LOW**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Chr12a	IA-PM-PR	TS-PM-PR	GA-PM-PR	GA-RK-PR	SA-RK-PR	SA-PM-PR	IA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**LOW (N)**	**LOW**
Chr12b	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	SA-RK-PR	SA-PM-PR	IA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**LOW (N)**	**LOW**
Chr12c	TS-PM-PR	IA-PM-PR	GA-PM-PR	SA-RK-PR	GA-RK-PR	SA-PM-PR	IA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**MODERATE (N)**	**LOW**
Chr15a	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	SA-PM-PR	IA-RK-PR	SA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**MODERATE (P)**	**LOW**

Name	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	Rank 8	FD	DD	Sparsity	Correlation N: Negative P: Positive	Problem Size
Chr15b	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**LOW (P)**	**LOW**
Chr15c	TS-PM-PR	GA-PM-PR	IA-PM-PR	GA-RK-PR	SA-PM-PR	IA-RK-PR	SA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**LOW (P)**	**LOW**
Chr18a	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**LOW (N)**	**LOW**
Chr18b	IA-PM-PR	TS-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**MODERATE (N)**	**LOW**
Chr20a	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**LOW (P)**	**LOW**
Chr20b	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**MODERATE (N)**	**LOW**
Chr20c	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**LOW (P)**	**LOW**
Chr22a	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-RK-PR	SA-PM-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**LOW (N)**	**LOW**
Chr22b	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**LOW (P)**	**LOW**
Chr22c	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**LOW (P)**	**LOW**
Els19	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	SA-PM-PR	SA-RK-PR	IA-RK-PR	TS-RK-PR	**HIGH**	**LOW**	**MODERATE**	**MODERATE (N)**	**LOW**
Esc16a	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	SA-RK-PR	SA-PM-PR	IA-RK-PR	TS-RK-PR	**MODERATE**	**MODERATE**	**LOW**	**LOW (N)**	**LOW**
Esc16b	ALL	ALL	ALL	ALL	ALL	ALL	ALL	ALL	**MODERATE**	**MODERATE**	**LOW**	**MODERATE (P)**	**LOW**
Esc16c	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	IA-RK-PR	SA-RK-PR	SA-PM-PR	TS-RK-PR	**MODERATE**	**MODERATE**	**LOW**	**MODERATE (N)**	**LOW**
Esc16d	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	SA-RK-PR	SA-PM-PR	IA-RK-PR	TS-RK-PR	**MODERATE**	**HIGH**	**MODERATE**	**MODERATE (P)**	**LOW**
Esc16e	GA-PM-PR	IA-PM-PR	TS-PM-PR	SA-PM-PR	SA-RK-PR	GA-RK-PR	IA-RK-PR	TS-RK-PR	**MODERATE**	**HIGH**	**MODERATE**	**LOW (N)**	**LOW**
Esc16f	**ALL**	**ALL**	**ALL**	**ALL**	**ALL**	**ALL**	**ALL**	**ALL**	**MODERATE**	**HIGH**	**MODERATE**	**LOW (P)**	**LOW**
Esc16g	GA-RK-PR	GA-PM-PR	IA-PM-PR	TS-PM-PR	SA-RK-PR	SA-PM-PR	IA-RK-PR	TS-RK-PR	**MODERATE**	**HIGH**	**MODERATE**	**MODERATE (N)**	**LOW**
Esc16h	GA-RK-PR	SA-RK-PR	GA-PM-PR	IA-PM-PR	TS-PM-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	**MODERATE**	**MODERATE**	**LOW**	**MODERATE (P)**	**LOW**
Esc16i	GA-RK-PR	SA-RK-PR	GA-PM-PR	IA-PM-PR	TS-PM-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	**MODERATE**	**HIGH**	**MODERATE**	**LOW (N)**	**LOW**

Name	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	Rank 8	FD	DD	Sparsity	Correlation N: Negative P: Positive	Problem Size
Esc16j	GA-RK-PR	SA-RK-PR	GA-PM-PR	IA-PM-PR	TS-PM-PR	SA-PM-PR	IA-RK-PR	TS-RK-PR	**MODERATE**	**HIGH**	**MODERATE**	**LOW (N)**	**LOW**
Esc32a	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**LOW**	**HIGH**	**MODERATE**	**LOW (N)**	**LOW**
Esc32b	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**LOW**	**HIGH**	**MODERATE**	**MODERATE (N)**	**LOW**
Esc32c	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**MODERATE**	**LOW (N)**	**LOW**
Esc32d	GA-PM-PR	TS-PM-PR	IA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**LOW**	**HIGH**	**MODERATE**	**LOW (N)**	**LOW**
Esc32e	GA-RK-PR	IA-RK-PR	SA-RK-PR	GA-PM-PR	IA-PM-PR	TS-PM-PR	SA-PM-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**LOW (N)**	**LOW**
Esc32f	GA-RK-PR	SA-RK-PR	GA-PM-PR	IA-PM-PR	TS-PM-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	**LOW**	**HIGH**	**HIGH**	**LOW (N)**	**LOW**
Esc32g	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**LOW**	**MODERATE**	**MODERATE**	**LOW (N)**	**LOW**
Esc64a	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**LOW**	**HIGH**	**HIGH**	**LOW (N)**	**MODERATE**
Esc128	TS-PM-PR	GA-PM-PR	IA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**LOW**	**HIGH**	**HIGH**	**MODERATE (P)**	**HIGH**
Had12	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	SA-PM-PR	IA-RK-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**LOW**
Had14	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	IA-RK-PR	SA-RK-PR	SA-PM-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**LOW**
Had16	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**LOW**
Had18	IA-PM-PR	TS-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**LOW**
Had20	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	TS-PM-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**LOW**
Kra30a	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**MODERATE**	**LOW**	**MODERATE**	**LOW (N)**	**LOW**
Kra30b	GA-RK-PR	IA-RK-PR	SA-RK-PR	TS-RK-PR	GA-PM-PR	IA-PM-PR	SA-PM-PR	TS-PM-PR	**MODERATE**	**LOW**	**MODERATE**	**LOW (N)**	**LOW**
Kra32	TS-PM-PR	GA-PM-PR	IA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**LOW**	**MODERATE**	**MODERATE**	**LOW (N)**	**LOW**
Lipa20a	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**LOW**
Lipa20b	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**HIGH (N)**	**LOW**

Name	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	Rank 8	FD	DD	Sparsity	Correlation N: Negative P: Positive	Problem Size
Lipa30a	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**LOW**
Lipa30b	TS-PM-PR	GA-PM-PR	IA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**HIGH (N)**	**LOW**
Lipa40a	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**MODERATE**
Lipa40b	TS-PM-PR	GA-PM-PR	IA-PM-PR	IA-RK-PR	GA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**HIGH (N)**	**MODERATE**
Lipa50a	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**MODERATE**
Lipa50b	TS-PM-PR	GA-PM-PR	IA-PM-PR	IA-RK-PR	GA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**HIGH (N)**	**MODERATE**
Lipa60a	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**MODERATE**
Lipa60b	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**HIGH (N)**	**MODERATE**
Lipa70a	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**MODERATE**
Lipa70b	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**HIGH (N)**	**MODERATE**
Lipa80a	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**HIGH**
Lipa80b	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**HIGH (N)**	**HIGH**
Lipa90a	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	TS-PM-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**HIGH**
Lipa90b	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**HIGH (N)**	**HIGH**
Nug12	IA-PM-PR	TS-PM-PR	GA-PM-PR	SA-PM-PR	GA-RK-PR	SA-RK-PR	IA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (N)**	**LOW**
Nug14	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Nug15	IA-PM-PR	TS-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Nug16a	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**MODERATE (P)**	**LOW**
Nug16b	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Nug17	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**MODERATE (P)**	**LOW**

Name	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	Rank 8	FD	DD	Sparsity	Correlation N: Negative P: Positive	Problem Size
Nug18	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**MODERATE (P)**	**LOW**
Nug20	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**MODERATE (P)**	**LOW**
Nug21	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Nug22	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Nug24	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Nug25	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Nug27	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**MODERATE**	**LOW**	**MODERATE (P)**	**LOW**
Nug30	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Rou12	TS-PM-PR	IA-PM-PR	GA-PM-PR	SA-PM-PR	GA-RK-PR	SA-RK-PR	IA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Rou15	IA-PM-PR	TS-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Rou20	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Scr12	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	SA-RK-PR	SA-PM-PR	IA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**MODERATE**	**LOW (N)**	**LOW**
Scr15	IA-PM-PR	TS-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**MODERATE**	**MODERATE (N)**	**LOW**
Scr20	IA-PM-PR	TS-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**HIGH**	**MODERATE**	**LOW (N)**	**LOW**
Sko42	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**MODERATE**
Sko49	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**MODERATE**
Sko56	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**MODERATE**
Sko64	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**MODERATE**
Sko72	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**MODERATE**
Sko81	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**HIGH**

Name	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	Rank 8	FD	DD	Sparsity	Correlation N: Negative P: Positive	Problem Size
Sko90	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**HIGH**
Sko100a	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**HIGH**
Sko100b	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	TS-RK-PR	SA-RK-PR	SA-PM-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**HIGH**
Sko100c	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**HIGH**
Sko100d	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**HIGH**
Sko100e	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (N)**	**HIGH**
Sko100f	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**HIGH**
Ste36a	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**HIGH**	**LOW**	**HIGH**	**LOW (N)**	**LOW**
Ste36b	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**HIGH**	**MODERATE**	**HIGH**	**LOW (N)**	**LOW**
Ste36c	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**HIGH**	**LOW**	**HIGH**	**MODERATE (N)**	**LOW**
Tai12a	IA-PM-PR	TS-PM-PR	GA-PM-PR	SA-PM-PR	GA-RK-PR	SA-RK-PR	IA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**MODERATE (P)**	**LOW**
Tai12b	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	SA-PM-PR	SA-RK-PR	IA-RK-PR	TS-RK-PR	**HIGH**	**MODERATE**	**LOW**	**MODERATE (P)**	**LOW**
Tai15a	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Tai15b	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**HIGH**	**HIGH**	**LOW**	**MODERATE (P)**	**LOW**
Tai17	IA-PM-PR	TS-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**LOW**
Tai20a	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Tai20b	IA-PM-PR	GA-PM-PR	TS-PM-PR	GA-RK-PR	SA-PM-PR	SA-RK-PR	IA-RK-PR	TS-RK-PR	**HIGH**	**MODERATE**	**LOW**	**LOW (N)**	**LOW**
Tai25a	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Tai25b	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**HIGH**	**MODERATE**	**LOW**	**LOW (P)**	**LOW**
Tai30a	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**LOW**

Name	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5	Rank 6	Rank 7	Rank 8	FD	DD	Sparsity	Correlation N: Negative P: Positive	Problem Size
Tai30b	IA-PM-PR	GA-PM-PR	TS-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**HIGH**	**MODERATE**	**LOW**	**LOW (P)**	**LOW**
Tai35a	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**LOW (P)**	**LOW**
Tai35b	GA-PM-PR	IA-PM-PR	TS-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**HIGH**	**MODERATE**	**MODERATE**	**LOW (N)**	**LOW**
Tai40a	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**MODERATE (P)**	**MODERATE**
Tai40b	IA-PM-PR	TS-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**HIGH**	**LOW**	**LOW**	**LOW (P)**	**MODERATE**
Tai50a	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**LOW (P)**	**MODERATE**
Tai50b	IA-PM-PR	TS-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**HIGH**	**MODERATE**	**MODERATE**	**LOW (N)**	**MODERATE**
Tai60a	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**LOW (P)**	**MODERATE**
Tai60b	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**HIGH**	**MODERATE**	**MODERATE**	**LOW (P)**	**MODERATE**
Tai64	IA-PM-PR	GA-PM-PR	GA-RK-PR	TS-PM-PR	SA-PM-PR	SA-RK-PR	IA-RK-PR	TS-RK-PR	**MODERATE**	**HIGH**	**HIGH**	**MODERATE (P)**	**MODERATE**
Tai80a	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**LOW (P)**	**HIGH**
Tai80b	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**HIGH**	**LOW**	**MODERATE**	**LOW (P)**	**HIGH**
Tai100b	TS-PM-PR	IA-PM-PR	GA-PM-PR	IA-RK-PR	GA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**LOW (P)**	**HIGH**
Tai100a	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	TS-RK-PR	SA-PM-PR	SA-RK-PR	**HIGH**	**MODERATE**	**MODERATE**	**LOW (P)**	**HIGH**
Tho30	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	SA-RK-PR	TS-RK-PR	**MODERATE**	**LOW**	**LOW**	**LOW (N)**	**LOW**
Tho40	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**MODERATE**	**LOW**	**MODERATE**	**LOW (P)**	**MODERATE**
Tho50	TS-PM-PR	IA-PM-PR	GA-PM-PR	GA-RK-PR	IA-RK-PR	SA-PM-PR	TS-RK-PR	SA-RK-PR	**LOW**	**LOW**	**LOW**	**LOW (P)**	**MODERATE**