The Quadratic Assignment Problem: Special Cases and Relatives

Eranda Çela

Dissertation
zur Erlangung des Titels eines Doktors
der Technischen Wissenschaften,
eingereicht an der Technisch-
Naturwissenschaftlichen Fakultät
der Technischen Universität Graz

Graz, im November 1995

Betreuer:
o.Univ.-Prof. Dr. Rainer E. Burkard

Adresse:
Institut für Mathematik B
Technische Universität Graz
Steyrergasse 30/II
A-8010 Graz
Austria

Financial support by the “Fonds zur Förderung der wissenschaftlichen Forschung, Projekt P8971-PHY”, by the “Christian Doppler Laboratorium für Diskrete Optimierung” and by the “Spezialforschungsbereich, Projektbereich Diskrete Optimierung” is gratefully acknowledged.

Prindërve të mi

Acknowledgments

First of all, I express my deepest gratitude to my supervisor Prof. Rainer E. Burkard for his manifold help and encouragement during the last three years. He motivated me to investigate the quadratic assignment problem (QAP) and its generalizations. I appreciate very much the opportunity to benefit from his long and rich experience with the QAP. Prof. Burkard's support was decisive for providing the financial basis which enabled my graduate studies.

Next, I would like to thank Franz Rendl for his willingness to be the second referee of this thesis.

I am really happy to have worked with Gerhard J. Woeginger to whom I would like to express my special thanks and admiration. The results presented in Chapters 3, 4 and 7 are fruit of our joint research. While working together, I could benefit from his outstanding knowledge of complexity theory in particular and of combinatorial optimization in general. I learned a lot from his clarity of thinking and also from his rigorous critique. Gerhard's confidence gave me invaluable help to hold onto my research work. Finally, I would like to thank Gerhard for his moral support just when I needed it most and for being a wonderful friend.

The next stream of thanks goes to Bettina Klinz for her constant help and friendship. Her encouragement and confidence helped me to cope with disappointments which seem to go hand in hand with research work. I highly appreciate our various discussions and Bettina's helpful suggestions.

My thanks go also to Günter Rote for helpful discussions and essential remarks especially when working on a joint paper on special cases of the QAP. These results are included in Chapter 3.

Furthermore, I owe special thanks to my office mates and good friends Rüdiger Rudolf and Ulrich Pferschy for numerous helpful discussions.

Finally, I would like to thank all my coauthors for their precious collaboration.

I am gratefully indebted to Federico Malucelli who kindly provided me with a copy of his thesis on quadratic assignment problems and for helpful discussions during my visit at the University of Pisa.

This research was done at the Institute of Mathematics B at the Graz University of Technology. The whole staff of the Institute and the colleagues of the Christian Doppler Laboratories have contributed to a nice and convenient working atmosphere. Thank you!

At this occasion, I cannot forget to express my gratitude to the staff of the Mathematics Department at the University of Tirana, Albania, where I completed my undergraduate studies. I highly appreciate their hard work despite the difficult conditions.

Special thanks are attributed to the "Fonds zur Förderung der wissenschaftlichen

Forschung (Projekt P8971-PHY)", the "Christian Doppler Laboratorium für Diskrete Optimierung" and the "Spezialforschungsbereich Optimierung und Kontrolle, Projektbereich Diskrete Optimierung", for the financial support while working on this thesis.

Last but not least, I would like to thank my family and my relatives for their love, confidence and support. Especially, I thank Nandi for his love and understanding, and for being there whenever I needed him. This thesis is dedicated to my parents for their infinite love and sacrifices, doing their best for me all my life long.

Abstract

In this thesis we consider the following three problems:

- The quadratic assignment problem (QAP)
- The biquadratic assignment problem (BiQAP)
- The communication assignment problem (CAP)

The thesis is organized in three parts, among which the first one gives a survey of the present state of the field and the other two contain new contributions.

In the first part some of the most significant results on QAPs are summarized and reviewed. As for the QAP aspects considered here, the following keywords can be mentioned: alternative formulations, computational complexity, lower bounds and exact algorithms, heuristics, QAPs with known optimal solution, asymptotic behavior.

In the second part special cases of the QAP are investigated from the point of view of computational complexity. We distinguish two groups of special cases. The first group consists of QAPs whose coefficient matrices have certain combinatorial properties. Among others, QAPs on Monge and Anti-Monge matrices, Toeplitz and circulant matrices, taxonomy matrices and matrices with limited bandwidth are considered. The second group consists of QAP formulations of well known problems in graphs such as placement problems, minimum feedback arc set problems and packing problems. Additionally, special cases of the so called *general quadratic assignment problem (GQAP)* are considered. Most of them arise in the context of isomorphic graphs. Trying to draw a borderline between "easy" and "hard" versions of the QAP, a number of polynomially solvable and $\mathcal{NP}$-hard cases are identified. A constructive analysis of methods and proof ideas is provided together with a set of open question as promising directions for further research.

In the third part of the thesis the BiQAP and the CAP are investigated. We introduce the BiQAP as a mathematical model for a problem of practical relevance arising in VLSI design. The BiQAP is a straightforward generalization of the QAP and contains the QAP as a special case. Generalizing similar approaches applied to QAPs, a generator of BiQAP instances with known optimal solution and lower bounds for the BiQAP are proposed. Moreover, the asymptotic behavior of this problem is investigated, showing that also in this aspect, the BiQAP's behavior is similar to that of the QAP. From the practical point of view, our main contribution concerning the BiQAP consists in proposing and testing several heuristics. Three versions of improvement methods, three versions of simulated annealing and two versions of tabu search are tested on BiQAP instances with known optimal solution. The tests aim mainly to compare the performance of different heuristics, but also

to (empirically) analyze the sensitivity of these methods towards different control parameters.

Finally, we introduce the CAP as a mathematical model for a problem arising when locating stations in a local area computer network (LAN), aiming to minimize the overall traffic going through the busiest station. The computational complexity of several versions of this problem with different network topologies is investigated. It turns out that in most cases the CAP remains $\mathcal{NP}$-hard. However, two polynomially solvable cases are singled out, where the underlying network is a star of branch length 2 or a doublestar. For the CAP on trees a branch and bound algorithm is proposed. This algorithm is tested on randomly generated instances of CAP where the underlying tree has up to 15 vertices. The investigation of the asymptotic behavior of the CAP on trees shows that, under natural probabilistic constraints, every feasible solution gets in some sense close to the optimal one, as the size of the problem increases.

Contents

Chapter 1

Introduction and Preliminaries

1.1 Motivation

The quadratic assignment problem (QAP) is widely considered as a classical combinatorial optimization problem, though being relatively young. A number of prominent researchers have been dealing with the QAP since it was introduced in 1957 by Koopmans and Beckmann to model a plant location problem. In our opinion there are three main reasons which make the QAP extremely attractive.

First, the number of real-life problems which are mathematically modeled by QAPs has been continuously increasing and the variety of the fields they belong to is astonishing. To recall just a restricted number among the applications of the QAP let us mention placement problems, scheduling, manufacturing, VLSI design, statistical data analysis, and parallel and distributed computing.

Secondly, a number of other well known combinatorial optimization problems can be formulated as QAPs. Typical examples are the traveling salesman problem and a large number of problems in graphs such as the maximum clique problem, the graph partitioning or the minimum feedback arc set problem.

Finally, from the computational point of view the QAP is one of the most difficult problems to solve; solving instances of size larger than 20 is nowadays still considered intractable. Notice that the remarkable progress in data structures and algorithmic developments, as well as major advances in computer hardware, have enabled a tremendous increase in the size of NP-hard problems which can be solved in practice. Consider for instance large-scale instances of combinatorial problems such as the maximum clique problem on graphs with thousands of vertices and millions of edges, or the traveling salesman problem with thousands of cities. In this context, the quadratic assignment problem remains a challenging exception.

Despite the toughness of the QAP and also because of it, the literature on this problems abounds in results on almost all usually considered aspects of combinatorial optimization problems. However, whereas a large number of papers have been written, for example, on bounding techniques and heuristics for the QAP, the complexity of restricted versions of the problem seems to be less studied. Moreover, the numerous well solvable cases of the TSP, which is itself a special case of the QAP, encourage research efforts in this direction. These observations motivated the identification of new polynomially solvable and provably hard cases of the QAP in particular, and the systematization and classification of already existing results in a unique framework in general. Two groups of restricted versions of the QAP are considered in this thesis, where the restrictions either concern the combinatorial structure of the QAP coefficient matrices or are related to QAP formulations of well known problems in graphs. Among others, QAPs on special matrix classes such as Monge, Anti-Monge, Toeplitz, circulant and taxonomy matrices have been investigated, as well as QAPs related to placement problems, minimum feedback arc set problems and packing problems in graphs. Polynomiality and NP-completeness results on these QAPs constitute a considerable part of our contribution.

Further on, motivated by a problem arising in VLSI design, we introduce and investigate a generalization of the QAP, the biquadratic assignment problem (BiQAP). The BiQAP contains the QAP as a special case. Similarly as the QAP, BiQAP is quite simple to state but very difficult to deal with. Generalizing results which are already known for QAPs, a bounding procedure and a generator of instances with known optimal solution are proposed. Moreover, the probabilistic asymptotic behavior of the BiQAP is investigated and heuristics for its solution are proposed and computationally tested. However, these are only the first steps towards efficiently dealing with the problem and many open questions remain as topics for future research.

Finally, we consider the communication assignment problem (CAP). The motivation for dealing with this new problem is, similarly as for BiQAPs, of practical nature. The problem arises when locating stations (computers, terminals) in a local area computer network, trying to minimize the traffic going *through* the busiest station. A detailed analysis of the computational complexity of different versions of the problem with different topologies of the underlying network, shows that in most cases the CAP remains NP-hard. This behavior suggests that implicit enumeration methods are the only potential mean to solve the problem. Hence, we propose and describe a branch and bound algorithm for the CAP, in the case that the underlying network is a tree. Again, this is only the first effort to solve the CAP and a lot remains to be done in order to improve the quality of the involved lower bounds and decrease their computation time. Despite of its inherent complexity, the CAP on trees shows an interesting asymptotic behavior. Namely, under natural probabilistic constraints, the problem becomes almost trivial in the sense that every feasible solution is somehow close to the optimal one and shares this property with the closely related QAP.

1.2 Preview

This thesis starts with a preliminary section aiming to recall some of the main basic notions from combinatorial optimization, complexity theory and graph theory. We also try to unify the matrix and vector notation to be used throughout the thesis. Finally, the four *assignment problems* to be dealt with in this thesis are introduced: the quadratic assignment problem (QAP), the general quadratic assignment problem (GQAP), the biquadratic assignment problem (BiQAP) and the communication assignment problem (CAP).

The thesis is organized in three parts:

Part I The Quadratic Assignment Problem (QAP): A Short Overview.

Part II Easy and Hard Cases of the QAP: Drawing a Borderline.

Part III A Generalization of the QAP and Related Problems.

The first part is introductory and not really related to the second one, whereas to some extent related to the third one. Considerable efforts have been made to unify the presentation and the notation. The next section in this chapter should serve this goal, among others. However, for the sake of the readability, some of the definitions and notations are reintroduced in the chapters where they occur. As the three parts of the thesis are to some extent independent and as our contribution concerns different problems and problem aspects respectively, there is no common chapter of conclusions. In some of the chapters there is a concluding section providing a summary of results and open questions related to the corresponding considered problem. At the end of the thesis there is a common bibliography.

Let us shortly preview the contents of each part to get a general overview on the results presented in this thesis. For more details the reader is referred to the introductory sections in the respective parts and chapters.

Part I

This part consists of one chapter, namely Chapter 2, and provides a short overview on the *quadratic assignment problem (QAP)*. As the main part of our contribution concerns either aspects of the QAP itself or problems somehow related to it, we decided to include such an overview in this thesis. Since 1957 when the QAP was originally introduced, the literature abounds in studies and results on this problem. The diversity of the work done on QAPs makes probably interesting the visualization of the main streams of results in order to better understand the current state of the research, the new and old open problems, and to correctly locate our contribution in this research area. After having introduced the QAP and having mentioned some of its numerous applications, some of the most significant results concerning computational complexity, exact algorithms and heuristics for QAPs, are summarized.

Further on, the generation of QAP instances with known optimal solution and the asymptotic behavior of the QAP are discussed. The results related to the two last aspects can be generalized also for BiQAPs, as shown in the third part of the thesis.

Part II

In this part of the thesis the challenging problem of distinguishing polynomially solvable cases of the QAP is considered. We try to summarize what is known about the complexity of restricted versions of the QAP, where the restrictions either concern the structure of the coefficient matrices or are related to QAP formulations of well known problems in graphs. Additionally, we formulate a set of open questions on QAPs with unknown complexity, which we consider to be a consistent and promising object of further research in this direction.

This part consists of two chapters. In Chapter 3 we present results on versions of the QAP whose coefficient matrices are specially structured, i.e., have certain combinatorial properties. The so called *constant QAPs* and *constant permutation QAPs* are introduced. QAPs for which all feasible solutions yield the same objective function value are termed constant QAPs. A restricted version of the QAP whose instances of an arbitrary fixed size have a common optimal solution is termed a constant permutation QAP. Such versions of the QAP are easy to solve, though the proofs of their polynomiality are often complicated and rather technical. It turns out that most of the polynomially solvable restricted versions of the QAP singled out in this thesis belong to one of these groups.

In Sections 3.2, 3.3, 3.4 and 3.5, QAPs on Monge and Anti-Monge matrices (Sum, Product, Large and Small matrices in particular), taxonomy matrices, Toeplitz, circulant and limited bandwidth matrices are investigated. Several polynomially solvable cases of the problem are singled out and many others are proved to be NP-hard.
These sections are partly based on a working paper on special cases of the QAP, coauthored by R. E. Burkard, V. M. Demidenko, N. N. Metelski and G. J. Woeginger.

The results presented in Sections 3.6 and 3.7 concern the QAP with an Anti-Monge matrix and a Toeplitz matrix. Here, the main result is the identification of three properties of the Toeplitz matrix which make the above mentioned QAP polynomially solvable, or more precisely, a constant permutation QAP. Additionally it is proven that this version of the QAP is in general NP-hard, answering at the same time the interesting question about the complexity of the so called "turbine problem". Finally two further applications of this problem are described.
The results presented in these sections are taken from the paper "The Quadratic Assignment Problem with an Anti-Monge and a Toeplitz matrix: Easy and Hard Cases", SFB Report Nr. 34, University of Technology Graz, coauthored by R. E. Burkard, G. Rote and G. J. Woeginger.
The chapter is completed by a concluding section which provides a summary of results on QAPs with specially structured coefficient matrices. Moreover, the proof methods

for the polynomiality results are analyzed and the next-to-solve open questions are singled out and discussed.

In Chapter 4 QAP formulations of some well known optimization problems in graphs such as placement problems, feedback arc set problem and packing problems, are considered. We show how the results on polynomially solvable and NP-hard versions of these graph problems can be translated in the QAP language. This investigation gives rise to new polynomially solvable and provably hard cases of the QAP, respectively. This chapter contains also some results on special cases of the GQAP. These special cases arise in the context of isomorphic graphs and that is why they are included in this chapter.
Also this chapter is partly based on the above mentioned working paper on special cases of the QAP.

Part III

In this part of the thesis we introduce and investigate a generalization of the QAP, namely *the biquadratic assignment problem* (BiQAP) and another assignment problem, the so called *communication assignment problem* (CAP). This part is organized in three chapters. In Chapter 5 the BiQAP is introduced as a mathematical formulation of a problem arising in VLSI design. Among the results presented in this chapter two can be singled out as the most important ones: a bounding procedure and an algorithm for generating BiQAP instances with known optimal solution. Both, the bounding procedure and the generator of instances with known optimal solution are generalizations of similar approaches for QAPs. Additionally, the probabilistic asymptotic behavior of the BiQAP has been investigated. We show that, under natural probabilistic constraints, BiQAPs of large size become somehow trivial. Namely, the ratio between the "best" and the "worst" objective function values approaches 1, with probability tending to 1, as the size of the problem approaches infinity.
This chapter is based on the paper "On the Biquadratic Assignment Problem", which appeared in *Quadratic Assignment and Related Problems*, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **16**, 1994, with coauthors R. E. Burkard and B. Klinz.

In Chapter 6 several heuristics for BiQAPs are proposed and tested. We investigated three versions of deterministic improvement methods, three versions of simulated annealing and two versions of tabu search on BiQAPs. The heuristics are tested on BiQAP instances with known optimal solutions. A detailed analysis of the performance of these methods and of their sensitivity versus different values of the corresponding control parameters is presented. One version of simulated annealing, which shows the best tradeoff between computation time and solution quality, deserves special attention.
The results presented in this chapter are taken from the paper "Heuristics for biquadratic assignment problems and their computational comparison", which ap-

peared in *European Journal of Operational Research* 83, 1995, 283-300, coauthored by R. E. Burkard.
The chapter is then completed by a concluding section which summarizes the main results of both theoretical and practical relevance respectively, and provides a critical overview on some open problems to be probably considered in the near future.

In Chapter 7 the communication assignment problem is introduced. This problem arises when locating stations in local area computer networks. Several version of this problem are distinguished with respect to the topology of the network to which the stations should be located. It is proven that even for very simple networks like paths and cycles the corresponding problem is NP-hard. Two polynomiality results are also derived for very special underlying networks such as stars of branch length two and double stars. The rest of the results presented in this chapter concerns the communication assignment problem on trees. The analysis of the probabilistic asymptotic behavior of this problem shows similar results as in the case of QAPs and BiQAPs. Namely, under natural probabilistic constraints, the ratio between the best and the worst objective function values approaches 1 with probability tending to 1 as the size of the problem approaches infinity. From the practical point of view, the main result in this Chapter 7 is a branch and bound algorithm for the communication assignment problem on trees, which is tested on randomly generated instances of the problem. It turns out that the expensive lower bounds and their poor quality in the first stages of the search hurt the robustness of the method.
The results presented in this chapter are taken form the paper "A Minimax Assignment Problem in Treelike Communication Networks", to appear in *European Journal of Operational Research*. This paper is coauthored by R. E. Burkard and G. J. Woeginger.
The chapter is completed by a review of results on the communication assignment problem, a summary of open questions and some concluding remarks.

1.3 Preliminaries

In this section we introduce the most important mathematical concepts and tools to be used throughout the thesis. The depth of the discussion corresponds to the goal: we just want to recall some basic definitions and make a few general comments about topics to be referred to on and on through the thesis.

We consider combinatorial optimization problems and treat them as such. So, we start this preliminary section with a formal definition of combinatorial optimization problems and a very short look at them. Following this topic we discuss some main aspects of local search. Local search is an important approach toward the practical solution of difficult combinatorial optimization problems which is also applied to the *biquadratic assignment problem*. Some of the heuristic approaches for this problem, proposed in Chapter 6 are local search based procedures.

As a next step we give a short informal introduction in complexity theory. We decided to do so based on the following two reasons. First, the objective of the second part of the thesis is to distinguish polynomially solvable and provably hard cases of the quadratic assignment problem. To this extent, we prefer to previously summarize the basic notions and tools we are going to use for our proofs in Chapters 3 and 4. Secondly, we find that the most intriguing property of QAP is its inherent difficulty and we would like to provide a more or less complete description of these aspects of the problem. It is widely accepted that, in a theoretical level, such a description may be only given by means of the complexity theory.

Then, we go on by recalling some basic notions in graph theory and by introducing some graph classes, to be used both as tools and background for the results presented in the comming chapters. Subsequently, a few standard and non standard notations on matrices and vectors are introduced, in order to prevent eventual misunderstandings. Finally, we merely give the correct mathematical formulations of the four main problems to be dealt with in this thesis.

1.3.1 Combinatorial optimization

There is no general agreement upon the definition of a *combinatorial optimization problem*. Obviously, we neither can give an exact definition for *combinatorial optimization* as a research area. However, most of the scientists which have been working on this field seem to agree about the fact stated below:

What is nowadays referred to as *combinatorial optimization* derives from the combination and cross-fertilization of various research streams, such as Graph Theory, Integer Programming, Combinatorics and Discrete Mathematics.

Here, we would like to give a formal definition of a combinatorial optimization problem, which at least includes the combinatorial optimization problems mentioned and considered as such in this thesis. Let us begin by introducing optimization problems in general. *Optimization problems* concern themselves with the choice of a "best" configuration among a set of feasible ones in order to achieve some goal. An instance of an optimization problems could be generally defined as follows:

Definition 1.1 *An* instance of an optimization problem *is a pair $(\mathcal{F}, c)$ where $\mathcal{F}$ is any set, domain of the feasible solutions, and the mapping $c: \mathcal{F} \to \mathbb{R}$ is the objective (or cost) function. In the case of a minimization problem the goal is to find an $X \in \mathcal{F}$ for which $c(X) \le c(Y)$, for all $Y \in \mathcal{F}$. In the case of a maximization problem the goal is to find an $X \in \mathcal{F}$, such that $c(X) \ge C(Y))$, for all $Y \in \mathcal{F}$.*
Such a solution X is called globally optimal solution*, or, when no confusion can arise, simply an* optimal solution*.*

Definition 1.2 *An* optimization problem *P is a set of instances of an optimization problem.*

So, we distinguish between a *problem* and an *instance of a problem*: Informally, in an *instance* we are given the "input data" and have enough information to obtain a solution, whereas a *problem* is a collection of instances, usually all generated in a similar way.

Two main categories of optimization problems can be distinguished: problems with *continuous* variables and problems with *discrete* variables. In the first group of optimization problems we are looking for a set of real numbers or even a function, whereas in the second category of problems we are generally looking for an object which is taken from a discrete finite set, typically an integer, a set of integers, a permutation or a graph. This two kinds of problems generally have quite different flavors, and the methods for solving them have become quite diverse. Combinatorial optimization problems belong to the second category of problems as described above.

Definition 1.3 *Let a finite ground set $\mathcal{E}$ be given. Let $\mathcal{F}$ be a feasible solutions set consisting of subsets of $\mathcal{E}$, $\mathcal{F} \subseteq 2^{\mathcal{E}}$. Moreover, let f be a real cost function on $\mathcal{E}$, $f\colon \mathcal{E} \to \mathbb{R}$. A* combinatorial optimization problem *with* sum objective function *consists in finding a feasible solution $X_0 \in \mathcal{F}$, such that*

$$\text{(P)} \qquad \sum_{x\in X_0} f(x) = \min_{X\in\mathcal{F}} \sum_{x\in X} f(x) \quad \left(\sum_{x\in X_0} f(x) = \max_{X\in\mathcal{F}} \sum_{x\in X} f(x) \right)$$

P *is said to be a* minimization (maximization) *problem.*

A combinatorial optimization problem *with* bottleneck objective function *consist in finding a feasible solution $X_0 \in \mathcal{F}$, such that*

$$\text{(P}'\text{)} \qquad \max_{x\in X_0} f(x) = \min_{X\in\mathcal{F}} \max_{x\in X} f(x) \quad \left(\max_{x\in X_0} f(x) = \max_{X\in\mathcal{F}} \max_{x\in X} f(x) \right)$$

P′ *is said to be a* minimization (maximization) *problem.*

X_0 *is called a* globally optimal solution *to P (P′) and $\sum_{x\in X_0} f(x)$ is called the* optimal value *of P (P′). If no confusion arises X_0 is simply called* optimal solution. *An* instance of a combinatorial optimization problem *P is obtained by specifying two sets $\mathcal{E}$, $\mathcal{F}$ and a cost function f.*

From now on we are going to deal mostly with combinatorial optimization problems. So, we simply write "problem" and mean "combinatorial optimization problem", unless otherwise specified.

1.3.2 Local search

Consider an instance of a problem P obtained by specifying a ground set $\mathcal{E}$, a feasible solution set $\mathcal{F}$ and a cost function f. Let $X \in \mathcal{F}$ be a feasible solution. In many situations it is useful to define a set $\mathcal{N}(X)$ consisting of feasible solutions which are somehow close to X, $\mathcal{N}(X) \subset \mathcal{F}$. Such a set $\mathcal{N}(X)$ is called *neighborhood* of X. If, for example, $\mathcal{F} = \mathrm{I\!R}$, the set of points within a fixed Euclidean distance provides a natural neighborhood. Generally, the choice of $\mathcal{N}$ depends critically on the structure of $\mathcal{F}$.

Finding a globally optimal solution to a problem can be prohibitively difficult, whereas it could be possible to find a solution X' which is the "best" in its neighborhood $\mathcal{N}(X)$. Such a solution is called *locally optimal solution*.

Definition 1.4 *Consider an instance of a minimization (maximization) problem P with ground set $\mathcal{E}$, feasible solutions set $\mathcal{F}$ and cost function f. Moreover, let $\mathcal{N}$ be a neighborhood. A feasible solution X', $X' \in \mathcal{F}$, is called* locally optimal solution *to P with respect to $\mathcal{N}$, (or simply* locally optimal solution *whenever $\mathcal{N}$ is understood by context), if*

$$\sum_{x \in X'} f(x) \leq \sum_{y \in Y} f(y), \quad \textit{for all } Y \in \mathcal{N}(X')$$

$$\left(\sum_{x \in X'} f(x) \geq \sum_{y \in Y} f(y), \quad \textit{for all } Y \in \mathcal{N}(X') \right)$$

In general, a locally optimal solution is not globally optimal. However there exist situations when a locally optimal solution is also globally optimal. As an example, consider linear programming (formulated as combinatorial optimization problem). Let v be any vertex of the convex set of its feasible solutions. Define the neighborhood of v to be the set of all other vertices which can be obtained by applying to v a simplex pivot. If for a certain vertex v no improving (feasible) pivot exists, v is a locally optimal solution, according to the above definition. It is well known that in this case vertex v is also a globally optimal solution.

As mentioned above, sometimes it is reasonable and convenient to look for local optima rather than for global optima. The algorithms that do this job, i.e. whose output is a locally optimal solution, but not necessarily a global optimum to the problem given as input, are widely called *local search* algorithms. Let us introduce a generic local search algorithm. For simplicity, let us consider a minimization problem. (For maximization problems every thing works quite similarly.) Basically, each local search algorithm makes use of an *improving* subroutine which can be generally defined as below:

$$improve(X) = \begin{cases} \text{any } X_1 \in \mathcal{N}(X) \text{ with } \sum_{x \in X_1} f(x) < \sum_{x \in X} f(x), \text{ if such an } X_1 \text{ exists} \\ \text{"loc.opt." otherwise} \end{cases}$$

A generic local search algorithm is shown in Figure 1.1. It starts at some initial feasible solution $X \in \mathcal{F}$ and uses the subroutine *improve* to search for a better solution in its neighborhood. So long as an improved solution exists, the current solution is updated and the neighborhood search is repeated for the new solution. Thus, the algorithm stops when it reaches a locally optimal solution. The application

```
procedure local search
begin
$X$ := some initial starting point in $\mathcal{F}$;
while $improve(X) \neq$ "loc.opt." do
$X := improve(X)$;
return $X$
end
```

Figure 1.1: The general local search algorithm

of this algorithm involves a number of choices. First, we have to decide how to choose a feasible solution for starting with. Sometimes, it is practical to execute local search starting from several different solutions. In this case, we must again decide how many feasible solutions to choose as starting points and how to distribute them in $\mathcal{F}$. Secondly, we must choose a "good" neighborhood structure for the problem at hand and a method for searching it. It is beyond the scope of this thesis to generally analyze the criteria for distinguishing a "good" neighborhood structure from a bad one particularly, and to present a detailed theory of local search generally. The reader interested in theoretical aspects of local search is referred to [139, 165]. Some results concerning applications of local search algorithms to QAPs are presented in Section 2.5 in this chapter. Moreover, in Chapter 6 some local search approaches for BiQAPs are described. We only remark here that there is a clear trade-off between small and large neighborhoods. One hopes that a larger neighborhood provides better local optima, whereas taking a longer time to search. A smaller neighborhood, would generally provide worse local optima, but one can generate more of them within a fixed amount of running time. Should we generate fewer "stronger" local optima or more "weaker" ones? This and similar questions are usually answered empirically. Usually, local search approaches are guided by intuition and experiments, because very little theory is available as a guide. The design of effective local search algorithms is very much an art.

It is worthy to mention here that the local search is only one of the abounding approaches towards difficult problems. As an optimal solution to such problems can often not be found in a reasonable amount of time, the use of heuristics becomes indispensable. The solutions provided by heuristics, which either are empirically proven to be, or are hoped to be acceptable in terms of quality, are usually called *suboptimal solutions*.

1.3.3 Computational complexity

In this section we would like to introduce some basic concepts and standard notions from the complexity theory. In the main lines we follow the seminal work of Garey and Johnson [68]. Of course, it is beyond the scope of this thesis to give a mathematically rigorous introduction to the complexity theory. Consider however that a major contribution of this thesis consists in distinguishing "easy" and provably "hard" cases of QAP. Therefore, an informal but clear definition of concepts to be used throughout the thesis is an important matter. Thus, we think that the following subsection will be of benefit.

Problems and algorithms

Usually, in complexity theory *problems* and *problem instances* are distinguished. Informally speaking, a *problem* is a general question to be answered which usually possesses several formal parameters whose values are left unspecified. A problem is specified by giving a general description of all its parameters and a statement of properties the answer, or *solution*, is required to satisfy. An *instance of the problem* is obtained by specifying particular (feasible) values for all problem parameters. As an example, consider *the quadratic assignment problem* of size n, shortly denoted by QAP(A,B), where A and B are $n \times n$ coefficient matrices. The problem consists of finding a permutation π_0 of $\{1, 2 \ldots, n\}$ which minimizes the following double sum

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i),\pi(j)} b_{ij} ,$$

over all permutations π of $\{1, 2, \ldots, n\}$. The parameters of this problem are its size n and the matrices A and B, i.e., their entries a_{ij} and b_{ij}, $1 \le i, j \le n$. π_0 is an *(optimal) solution* to this problem. An instance of this problem is given by specifying n, A and B, for example $n = 3$,

$$A = \begin{pmatrix} 0 & 1 & 3 \\ 2 & 2 & 4 \\ 3 & 4 & 5 \end{pmatrix} \qquad B = \begin{pmatrix} 5 & 7 & 8 \\ 6 & 8 & 9 \\ 8 & 9 & 9 \end{pmatrix}$$

An optimal solution to this QAP instance is permutation $\pi \in \mathcal{S}_3$ given by $\pi(1) = 3,\ \pi(2) = 2, \pi(3) = 1$.

An *algorithm* for a certain problem is a step by step procedure which, given any instance of this problem, generates a solution or shows that no solution exists.

Problem size and time complexity

Usually, we are interested in finding the most "efficient" algorithm for a given problem. The term "efficient" is related to all kind of computing resources needed to execute the algorithm. Normally, however, by "most efficient" is meant *the fastest* algorithm. One reason to do so is that the time requirements are a dominant factor deciding whether an algorithm is good enough to be used in practice or not. On the other side, it is natural to expect that for a given algorithm, these time requirements depend on the "complexity" of the considered instance of the problem. Moreover, we would expect somehow the "complexity" of the problem instance to roughly grow as the *size* of the problem instance grows. Therefore, the time requirements of an algorithm are realistically expressed as a function of the size of problem instances. This function gives in this case a measure of the complexity of the problem instance. *The size* of a problem instance reflects in some sense the amount of input data needed to describe this instance. Of course, this amount depends on the *encoding* chosen for representing the input data. Usually, *binary* encodings are used. In this case the size of a problem instance equals the number of bits needed to represent it. For example the size of a QAP instance is a polynomial function of the problem size n and of $\log MAX$ where MAX is the magnitude of the largest entry of the coefficient matrices A and B, i.e. $MAX = \max\{|a_{ij}|, |b_{ij}|\colon\ 1 \le i, j \le n\}$[1].

The time complexity taken by an algorithm Υ for a certain problem is a function $t_v\colon \mathbb{N} \to \mathbb{N}$ which maps each natural number n to the maximal time $t_\Upsilon(n)$ needed by algorithm Υ for solving a problem instance of size n. Often, the time complexity is also called *running time* of the algorithm.

The time complexity can be measured in several ways. Here, we use the *unit time model* instead of the Turing machine model. The latter though being a rigorous mathematical model, is much more complicated than the unit time model. In the unit time model the time complexity of an algorithm is measured by the number of elementary arithmetic operations (i.e. additions, subtractions, multiplications, divisions and comparisons) the algorithm performs.

Polynomial, strongly polynomial and exponential time algorithms

We emphasized above that an important question related to an algorithm concerns its efficiency. Recall that we accepted the time complexity taken by an algorithm as a measure of its efficiency. For simple and less simple reasons, an algorithm whose time complexity grows "slowly" with the size of problem instances is considered more efficient than an algorithm with fast growth of its time complexity function. Roughly speaking, the expressions "slow" and "fast" are usually translated in a mathematical language as "polynomial" and "exponential", respectively.

An algorithm is said to be *polynomial* if its time complexity function is bounded from above by a polynomial in the size of the input. For example, the time complexity function of a polynomial algorithm for solving QAP(A,B) would be bounded from above by a polynomial of the problem size n and of $\log MAX$.

An algorithm is said to be *strongly polynomial* if its time complexity function is bounded from above by a polynomial in the number of input numbers, but does not

[1]However, when writting "the size of QAP(A,B) is n" we mean that its coefficient matrices A and B are $n \times n$ matrices, unless otherwise specified. Whenever considering the size of a QAP instance from the complexity theory point of view, thus as defined above, this will explicitly specified.

depend on the magnitude of the input data. For example, the time complexity function of a strongly polynomial algorithm for solving QAP(A,B) would be bounded from above by a polynomial which depends only on the problem size n. In order to emphasize the difference between strongly polynomial algorithms and polynomial algorithms, the latter are often called *weakly* polynomial algorithms. It is understandable that strongly polynomial algorithms are preferred to (weakly) polynomial algorithms, as their time requirements are not affected by the magnitude of the problem data.

Usually, an algorithm is said to be *exponential* if its time complexity function cannot be bounded from above by a polynomial in the size of the input. An important subclass of the class of exponential algorithms is the class of *pseudopolynomial* algorithms. An algorithm is said to be *pseudopolynomial* if its time complexity function is bounded from above by a polynomial in the number of input numbers and in the magnitude of the largest of the input numbers. In the case of QAP(A,B), the time complexity of a pseudopolynomial algorithm would be bounded from above by a polynomial in n and MAX, where n and MAX are as specified above.

Finally, let us recall here the meaning of the frequently used notations $\Theta(g(n))$, $\Omega(g(n))$, $O(g(n))$, for a function $g: \mathbb{N} \to \mathbb{N}$. Consider functions $f: \mathbb{N} \to \mathbb{N}$. The sets $\Theta(g(n))$, $\Omega(g(n))$ and $O(g(n))$ of functions mapping $\mathbb{N}$ to $\mathbb{N}$ are defined below:

$$\Theta(g(n)) = \Big\{ f(n)\colon \text{there exists } c_1, c_2 \geq 0 \text{ such that } c_1 g(n) \leq f(n) \leq c_2 g(n) \Big\}$$

$$O(g(n)) = \Big\{ f(n)\colon \text{there exists } c > 0 \text{ such that } f(n) \leq c g(n) \Big\}$$

$$\Omega(g(n)) = \Big\{ f(n)\colon \text{there exists } c > 0 \text{ such that } c g(n) \leq f(n) \Big\}$$

Instead of writting, for example, $f(n) \in O(g(n))$, we often write f *is* $O(g(n))$. We do not distinguish between these two types of notations throughout the thesis.

The problem classes $\mathcal{P}$, $\mathcal{NP}$ and $\mathcal{NP}$-complete

In order to simplify matters when distinguishing "easy" and "hard" problems, the complexity theory mainly deals with the so called *decision problems*. A decision problem is a problem which has only two possible exclusive solutions "yes" or "no". For a given decision problem the instances whose answer is "yes" are called yes-instances. Analogously, the instances whose answer is "no" are called no-instances. As an example, consider the following decision problem: "Is the optimal value of QAP(A,B) equal to zero?"

Trying to mathematically formalize somehow the expressions "easy problem" and "hard problem", the complexity theory distinguishes several classes among decision problems. The most important among these classes are the three classes informally defined below.

The class of all decision problem for which a polynomial time algorithm exists is called *the class* $\mathcal{P}$.

Before giving the definition of the class $\mathcal{NP}$ we must (informally) introduce the *nondeterministic algorithms.* A nondeterministic algorithm for a decision problem P has two stages. The first stage is a *guessing stage* and the second one is a *checking stage.* For a given instance I of P the guessing stage merely guesses some structure $S(I)$. Then, I and $S(I)$ are both provided as input to the checking stage of the algorithm. At this point, the checking stage proceeds by computing in a "normal" way and eventually halts with an answer "yes" or "no", or eventually never halts computing forever. A nondeterministic algorithm is said to *solve* a decision problem P, iff the following two assertions hold:

1. For each yes-instance I of P the guessing stage guesses a structure $S(I)$, which leads to an answer "yes" of the checking stage.

2. For any no-answer I of P there exists no structure S that, when guessed by the guessing stage will lead the checking stage to an answer "no".

The class $\mathcal{NP}$ consists of those decision problems which can be solved by polynomial nondeterministic algorithms.

A fundamental concept in complexity theory is the notion of *polynomial reductions.* A decision problem P_1 can be *polynomially reduced* to a decision problem P_2 iff, for every instance I_1 of P_1, an instance I_2 of P_2 can be constructed in polynomial time (with respect to the size of I_1), such that I_2 is a yes-instance if and only if I_1 is a yes-instance.

A decision problem P is said to be NP-*complete* if it fulfills the following two properties:

(i) P is a member of the class $\mathcal{NP}$.

(ii) All other problems in the class $\mathcal{NP}$ can be polynomially reduced to P.

In the case that P fulfills only (ii), but it is not known to be in $\mathcal{NP}$, then P is said to be an NP-hard problem.

Consider a problem P and a polynomial p in the number of input data of P. We will denote by P_p a problem consisting only of instances of P with the property that the *magnitude* of their input data is bounded above by the value of polynomial p for the input length. A problem P is called *NP-complete in the strong sense* if it belongs to $\mathcal{NP}$ and there exist a polynomial p such problem P_p as defined above is NP-complete.

NP-hard optimization problems

The complexity theory extends also to optimization problems. The key observation in these extension is the strong relationship between decision problems and optimization problems. Namely, each optimization problem can be naturally related to a decision problem. For example, consider QAP(A,B) as an optimization problem and a constant c_0. A decision problem naturally related to QAP(A,B) is the following: "Does it exist a permutation π (of appropriate size) such that $Z(A,B,\pi) \le c_0$?"

An *optimization problem* is said to be NP-*hard* iff the existence of a polynomial algorithm for solving it implies the existence of a polynomial algorithm for some NP-complete or NP-hard decision problem.

The complexity of local search

In the previous subsection we mentioned that local search is an alternative and sometimes promising approach to difficult problems. The reason is that finding a locally optimal solution is presumably "easier" than finding a global one. At this point a natural question arises: What means "easier" in the context of the complexity theory? The theoretical basis for facing this kind of problems is introduced by Johnson, Papadimitriou and Yannakakis in [95]. They define the so called *polynomial-time local search problems*, shortly *PLS problems*. Consider a pair $(P,\mathcal{N})$, where P is a combinatorial optimization problem P and $\mathcal{N}$ is a related neighborhood structure as defined in Subsection 1.3.2. Such a pair $(P,\mathcal{N})$ defines a *local search problem*, which consists in finding a locally optimal solution to P with respect to the neighborhood structure $\mathcal{N}$. An instance I of this local search problem is obtained by specifying the ground set $\mathcal{E}_I$, the feasible solutions set $\mathcal{F}_I$ and a cost function f_I. $(P,\mathcal{N})$ is said to be a *polynomial-time local search problem* if there exist three polynomial algorithms Υ_1, Υ_2 and Υ_3 having the following properties:

- For any instance I of P, algorithm Υ_1 produces a feasible solution $X_I \in \mathcal{F}_I$.
- For any instance I of P and a subset $X \subset \mathcal{E}_I$, algorithm Υ_2 checks whether $X \in \mathcal{F}_I$ and if $X \in \mathcal{F}_I$ it computes $\sum_{x\in X} f(x)$.
- If P is a minimization (maximization) problem, for any feasible solution $X \in \mathcal{F}_I$ of an instance I of P, algorithm Υ_3 produces a neighbor feasible solution $Y \in \mathcal{N}_I(X)$ with $\sum_{x\in X} f(x) > \sum_{y\in Y} f(y)$ ($\sum_{x\in X} f(x) < \sum_{y\in Y} f(y)$), if there is any. Otherwise it outputs that no such a solution exists and hence X is a locally optimal solution.

The class of PLS problems is denoted by $\mathcal{PLS}$. Thus, the class $\mathcal{PLS}$ consists essentially of those local search problems for which local optimality can be checked in polynomial time. In analogy to decision problems it can be shown the existence of complete problems in this class. Again, the notion of reduction, here called PLS-reduction, is crucial for the whole theory. A problem $(P_1,\mathcal{N}_1)$ is said to be $\mathcal{PLS}$-*reducible* to problem $(P_2,\mathcal{N}_2)$, iff the following three conditions hold:

(i) For every instance I_1 of P_1 an instance $I_2 = g(I_1)$ can be constructed in polynomial time.

(ii) For every instance I_1 of P_1 and for every feasible solution X_2 of the corresponding instance $g(I_1)$ of P_2, a feasible solution X_1 of I_1, $X_1 = h(X_2, I_1)$, can be constructed in polynomial time.

(iii) If X_2 is a locally optimal solution to $g(I_1)$, for some instance I_1 of P_1, then $h(X_2, I_1)$ is a locally optimal solution of instance I_1 of P_1.

A local search problem $(P, \mathcal{N})$ is said to be *PLS-complete* iff, first, $(P, \mathcal{N})$ is a member of class $\mathcal{PLS}$ and, secondly, all other problems in $\mathcal{PLS}$ are $\mathcal{PLS}$-reducible to $(P, \mathcal{N})$. The class of PLS-complete problems is denoted by $\mathcal{PLS}$-*complete*.

In [95] it is shown that the class $\mathcal{PLS}$-complete is not empty. Namely, the graph partitioning problem with the well known Kernighan-Lin neighborhood structure (defined in [103]) is a member of $\mathcal{PLS}$-complete. In the next chapter we will see that QAP with some appropriate neighborhood structures belongs also to this class.

1.3.4 Graph terminology and some classes of graphs

It is well known that the modeling of combinatorial optimization problems which arise in different research fields often involves graphs and notions form the graph theory. On the other side, a large number of optimization problems originally arise in a graphical context and thereby also involve notions and results from the graph theory. QAP and the other assignment problems related to it, to be discussed in this thesis, broadly inherit the relationship between combinatorial optimization and graph theory. Some traditional optimization and decision problems in graphs are inherently special cases of QAP. Due to their QAP formulations, these problems will turn out to be useful in distinguishing polynomially solvable cases of QAP in Chapter 4. To this extent, in this section we introduce definitions and notations from the graph theory to be used throughout the thesis. In our opinion, the benefit of this kind of introduction increases when taking into account the lack of a unified graph terminology in the up-to-date literature.

Graphs and digraphs

A *graph* $G = (V, E)$ consists of a finite, nonempty set of *vertices* V and a finite set E of *edges*. An *edge* is determined by 2 vertices which may also coincide. In the latter case the edge is called a *loop*. An edge defined by vertices i, j is denoted by (i, j). i and j are called *endvertices* of e. Given an edge $e = (i, j)$, we say that edge e and vertex i are *adjacent* to each other and so do edge e and vertex j. Vertices i and j are also said to be *adjacent* to each other. Two edges are called *parallel* if they share the same endvertices. A graph containing no loops and parallel edges is called a *simple graph*. As almost all graphs we consider are simple graphs, we generally use the term “graph” and mean a “simple graph”, unless otherwise specified. In a (simple) graph the *degree* of a vertex is the number of edges adjacent to it.

A *directed graph* or *digraph* $G = (V, E)$ consists of a finite nonempty set of *vertices* V and a finite set of *arcs* E. An *arc* is defined by an ordered pair of vertices, say i and j, and is denoted by (i, j). (It will be always clear from the context whether an arc or an edge is meant by (i, j).) Vertex i of arc $e = (i, j)$ is called the *tail* (or *startvertex*) of e and vertex j is called the *head* (or *endvertex*) of e. $e = (i, j)$ is called an *in-coming arc* for j and an *out-coming arc* for i. The *out-degree* of a vertex i is the number of arcs whose tail is i. The *in-degree* of a vertex i is the number of arcs whose head is i. The *degree* of a vertex is the sum of its in-degree and its out-degree. A vertex whose in-degree is equal to 0 is called a *source*, whereas a vertex whose out-degree is equal to 0 is called a *sink*. Two edges sharing the same tail and head, respectively, are called *parallel* edges. The adjacency definition and the definition of endvertices are here the same as for (undirected) graphs.

If numerical values are assigned to the (arcs) edges of a (directed) graph, the graph is called a *weighted* (directed) graph. The numerical values are usually termed *weights*. The *adjacency matrix* of a graph $G = (V, E)$ is a $|V| \times |V|$ matrix, say $A = (a_{ij})$, where $a_{ij} = 1$ if $(i, j) \in E$ and $a_{ij} = 0$ otherwise. The *weighted adjacency matrix* of a graph $G = (V, E)$ is a $|V| \times |V|$ matrix, say $A = (a_{ij})$, where a_{ij} equals the weight of edge (i, j), if $(i, j) \in E$ and $a_{ij} = 0$ otherwise. The (weighted) adjacency matrix of a digraph is defined analogously. If numerical values are assigned to the edges and/or vertices of a (directed) graph, then we get a (directed) *network*. In the following we do not make any difference between the terms "graph" and "network".

Let two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be given. G_1 and G_2 are said to be *isomorphic* if there exists a bijection $f\colon V_1 \to V_2$, such that $(i, j) \in E_1$ if and only if $(f(i), f(j)) \in E_2$.

Paths and cycles

A simple *path* P in a graph (digraph) $G = (V, E)$ is a sequence of pairwise distinct vertices $(v_1, v_2, \dots, v_k)$ such that $(v_i, v_{i+1}) \in E$, ($(v_i, v_{i+1}) \in E$ or $(v_{i+1}, v_i) \in E$ in the case of a digraph), for $1 \le i \le k-1$. We denote $P = (v_1, v_2, \dots, v_k)$. Vertex v_1 is *the startvertex* of path P and vertex v_2 is *the endvertex* of path P. v_1 and v_2 are called *endvertices* of P. $k-1$ is *the length* of the path P.

If $G = (V, E)$ is a digraph and either $(v_i, v_{i+1}) \in E$, for $1 \le i \le k-1$, or $(v_{i+1}, v_i) \in E$, for $1 \le i \le k-1$, then $P = (v_1, v_2, \dots, v_k)$ is called a *directed path* or *dipath*. A dipath with startvertex i and endvertex j is said to be a *dipath from i to j*.

We denote $(i, j) \in P$ if i,j are two consecutive vertices of (dipath) path P. If there exist only one path with startvertex i and endvertex j, this path is sometimes denoted by $P(i, j)$ (for example in a tree). For a vertex $v \in V$, the notation $v \in P$ means that v is a vertex in P different from both the startvertex and the endvertex. Such a vertex v is called an *inner vertex* of the path P.

A *cycle* is a path whose startvertex and endvertex coincide. If in the above definition the path is a dipath, then the corresponding cycle is called *dicycle*. The length of the (dipath) path in the above definition is also the *length of the cycle*. A cycle of length $|V|$ in a graph $G = (V, E)$ is a *Hamiltonian cycle* in G.

Some important graph classes

A graph $G = (V, E)$ is said to be *bipartite* if the vertex set V can be partitioned into two subsets V_1 and V_2, $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$, such that no edge in G has both its startvertex and endvertex in the same set V_i, $i = 1, 2$. In the case that $G = (V, E)$ is a digraph it is said to be *bipartite*, if each edge in E has its tail in V_1 and its head in V_2. Such a bipartite (digraph) graph is usually denoted by $G = (V_1, V_2, E)$.

A (digraph) graph is said to be *connected* if for each pair of vertices there exists a path having them as startvertex and endvertex, respectively. A digraph is said to be *strongly connected* if for each pair of vertices i, j, there exist a dipath from i to j and a dipath from j to i.
A *graph* is said to be *acyclic* if it contains no cycle. A *digraph* is said to be *acyclic* if it does not contain any dicycle. A connected and acyclic (digraph) graph is called a *tree*. A *leaf* in a tree is a vertex of degree 1. A tree is said to be a *rooted tree* if there exist a vertex r such that all paths connecting r with some other vertex v are dipaths from r to v. In this case r is called the *root* of the tree. Let (i, j) be an arc in a rooted tree. Vertex i is called *father* of j and vertex j is called *son* of i. The set of *successors* of a vertex i in a rooted tree G consists of those vertices j for which there exists a dipath from i to j in G.
A graph $G = (V, E)$ is called *complete graph* if $(i, j) \in E$, for each pair of vertices $i, j \in V$. The complete graph whose vertex set has cardinality n is denoted by K_n. A bipartite graph $G = (V_1, V_2, E)$ is said to be *complete bipartite graph* if for all $i \in V_1$ and for all $j \in V_2$, $(i, j) \in E$. If $|V_1| = n$ and $|V_2| = m$, the corresponding bipartite graph is denoted by $K_{n,m}$. Bipartite graphs of type $K_{1,n}$, $K_{2,n}$, $n \in \mathbb{N}$, are called *stars* and *doublestars*, respectively.
A *subgraph* of a (digraph) graph $G = (V, E)$ is a (digraph) graph $G' = (V', E')$ such that $G' \subseteq G$ and $E' \subseteq E$. Given a graph $G = (V, E)$ and a subset $V' \subset V$, the *subgraph of G induced by V* is a subgraph $G' = (V', E')$ defined by $E' = \{(i, j) \in E \colon i \in V',\ j \in V'\}$. Given an arbitrary graph G_0, a graph G is said to be a *G_0-free graph* if it does not contain G_0 as an induced subgraph. A *subdivision of a graph $G = (V, E)$* is a new graph $G' = (V', E')$ such that $G' = G \cup \{v_0\}$, for some $v_0 \notin G$ and there exists an edge $(i, j) \in E$ such that E' is given by $E' = E \setminus \{(i, j)\} \cup \{(i, v_0), (j, v_0)\}$.

A *flow graph* or a *rooted graph* is a digraph $G = (V, E)$ with a distinguished vertex r such that for each $v \in V \setminus \{r\}$ there exists a dipath from r to v. A *reducible flow graph* is a flow graph $G = (V, E)$ which can be reduced to a single vertex by a repeated application (in any order) of transformations T_1 and T_2 defined as below:

T_1 If there exists a $v \in V$ such that $(v, v) \in E$, remove (v, v) from E.

T_2 Let $v_2 \in V$ have a single in-coming arc (v_1, v_2). Contract the arc (v_1, v_2) by replacing vertices v_1, v_2 and arc (v_1, v_2) by a single vertex v. All arcs (x, v_1) are replaced by arcs (x, v) and all arcs (v_2, x) are replaced by (v, x). It results an arc (v, v) if there was formerly an arc (v_2, v_1) or (v_1, v_1).

A *regular graph of degree* x is graph all of whose vertices have degree x.
A graph is said to be *planar* if it can be drawn in the plane, with points representing its vertices and curves representing its edges, such that the endpoints of each curve represent the endvertices of the corresponding edge and the intersection of each pair of curves consists only of common endpoints. For a given graph with n vertices, it can be recognized in linear time, i.e., in $O(n)$ time, whether it is planar or not (see for example [92]).

A digraph is said to be *transitive* if for any two vertices i and j connected by a dipath from i to j, (i, j) is an edge. The *transitive closure* of a digraph $G = (V, E)$ is a digraph $G_T = (V, E_T)$ such that $(u, v) \in E_T$ if and only if either $(u, v) \in E$, or $u \neq v$ and there exists a dipath from u to v in G.

Concluding this section we introduce two more graph classes, the so called *vertex series-parallel* digraphs, shortly, *VSP* digraphs, and the *cographs*. The definition of this class is done in terms of its minimal members, the *minimal vertex series-parallel digraphs*, or shortly *MVSP digraphs* (see Valdes, Tarjan and Lawler [180]). *Minimal vertex series-parallel* (MVSP) digraphs are defined recursively as follows:

(i) The digraph containing a single vertex and no edges is a MVSP.

(ii) If $G_i = (V_i, E_i)$, $i = 1, 2$, are two vertex disjoint MVSP digraphs, then the digraphs constructed by the following operations are also MVSP:

- **a)** Parallel composition: $G_p := (V_1 \cup V_2, E_1 \cup E_2)$,
- **b)** Serial composition: $G_s := (V_1 \cup V_2, E_1 \cup E_2 \cup (T_1 \times S_2))$, where T_1 is the set of sinks of G_1 and S_2 is the set of sources of G_2.

A *vertex series-parallel digraph* is a digraph whose transitive closure equals the transitive closure of a MVSP digraph. It is usual to represent a MVSP by a rooted binary tree called the *decomposition tree*. The leaves of this tree are the vertices of the MVSP. The parallel (serial) decomposition of two MVSP digraphs G and G' is represented by a node labeled "P" ("S") with the roots of the decomposition trees of G and G' being its sons (cf. [180]). Contracting dipaths consisting only of "P" or "S" vertices into a single "P" or "S" vertex respectively, yields the so called *canonical decomposition tree* which is no longer binary.
In [180] it is proven that, given a digraph with n vertices and m arcs, it can be recognized in $O(n + m)$ time whether it is a VSP (MVSP) or not. Moreover, in the case that the considered digraph is MVSP, its decomposition tree can be constructed within the same amount of time.

There exist several equivalent definitions of cographs. We give here the recursive definition of them, which probably is not the most explicit one, but is less complicated in the sense that it does not involve additional concepts from graph theory. The cographs are defined recursively as follows:

(i) A graph containing a single vertex is a cograph.

(ii) If $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are cographs, then $G = (V_1 \cup V_2, E_1 \cup E_2)$ is a cograph, too.

(iii) If $G = (V, E)$ is a cograph, its complement $\bar{G} = (V, \bar{E})$ is also a cograph, where $\bar{E} = \{(v_i, v_j): 1 \leq i, j \leq n\ ,\ (v_i, v_j) \notin E\}$ and $V = \{v_1, v_2, \ldots, v_n\}$.

(iv) There are no other cographs.

Let us notice that the recognition of cographs can be done in linear time, see e.g. [47].

1.3.5 Matrix and vector notations

Here, we introduce some notations and definitions for vectors and matrices to be used throughout the thesis.

An $n \times m$ matrix A with entries a_{ij}, $1 \leq i \leq n$, $1 \leq j \leq m$, is denoted by $A = (a_{ij})$. A vector V with entries (v_i) is an $n \times 1$ matrix, for some $n \in \mathbb{N}$. It is denoted by $V = (v_i)$. Thus, all the operations on matrices defined below can be performed also on vectors.

The i^{th} column or row of matrix $A = (a_{ij})$ considered as vector is denoted by $A_{(i.)}$ or $A_{(.i)}$, respectively. The *transposed matrix* $A^t = (a^t_{ij})$ for an $n \times n$ matrix $A = (a_{ij})$ is defined by the equalities $a^t_{ij} = a_{ji}$, for $1 \leq i, j \leq n$.
An $n \times n$ matrix $A = (a_{ij})$ is said to be *symmetric* or *skew symmetric* if $a_{ij} = a_{ji}$ or $a_{ij} = -a_{ji}$, for $1 \leq i, j \leq n$, respectively. A matrix which is not symmetric is termed sometimes as *asymmetric.*
An $n \times n$ matrix $A = (a_{ij})$ is called a *Euclidean* matrix if its entries fulfill the triangle equality, i.e., $a_{ij} + a_{jk} \geq a_{ik}$, for any triple of indices $1 \leq i, j, k \leq n$.
An $n \times m$ matrix $A = (a_{ij})$ is called a *grid* matrix if it is the distance matrix of a grid on the vertices $v_1, v_2, \ldots, v_{nm}$, i.e., a_{ij} is the rectilinear (Manhattan) distance between the vertices v_i and v_j.
An $n \times n$ matrix $D = (d_{ij})$ with $d_{ij} = 0$ for all $i \neq j$ is called a *diagonal matrix* and is denoted by $D = diag(d_1, d_2, \ldots, d_n)$, where $d_i = d_{ii}$, $1 \leq i \leq n$.
Given two $n \times n$ matrices $A = (a_{ij})$, $B = (b_{ij})$ and two real numbers c, d the matrices $cA + d = (a'_{ij})$ and $A + B = (\bar{a}_{ij})$ are defined by the equalities $a'_{ij} = ca_{ij} + d$ and $\bar{a}_{ij} = a_{ij} + b_{ij}$, $1 \leq i, j \leq n$, respectively.

Given an $n \times n$ matrix $A = (a_{ij})$, an $n \times 1$ vector $V = (v_i)$ and a permutation π of $\{1, 2, \ldots, n\}$, the *permuted matrix* $A^\pi = (a^\pi_{ij})$ and the permuted vector $V^\pi = (v^\pi_i)$ are defined by the following equalities:

$$a^\pi_{ij} = a_{\pi(i)\pi(j)} \ , \ \text{ for } 1 \le i, j \le n \ , \quad \text{ and } \quad v^\pi_i = v_{\pi(i)} \ , \ \text{ for } 1 \le i \le n.$$

A permutation π of $\{1, 2, \ldots, n\}$ is sometimes specified by giving the sequence $\langle \pi(1), \pi(2), \ldots, \pi(n) \rangle$. The set of all permutations of $\{1, 2, \ldots, n\}$ is denoted by $\mathcal{S}_n$. The *identical permutation* is denoted by id, i.e. $id(i) = i$, $1 \le i \le n$. Given two permutations $\pi, \phi \in \mathcal{S}_n$, the product (or composition) of π with ϕ is a permutation $\psi \in \mathcal{S}_n$ defined by $\psi(i) = \pi(\phi(i))$, $1 \le i \le n$. We denote $\psi := \pi \circ \phi$. Often we say that *ψ is obtained by applying π to ϕ*. For some $n \in \mathbb{N}$, a *transposition* of $\{1, 2, \ldots, n\}$ is a permutation in $\mathcal{S}_n$ determined by a pair of indices $1 \le i, j \le n$, $i \ne j$. It is denoted by (i, j) and is defined as follows[2]:

$$(i,j)(k) = \begin{cases} k & k \notin \{i, j\} \\ i & k = j \\ j & k = i \end{cases}$$

A *partial permutation* of $\{1, 2, \ldots, n\}$ is a one to one mapping $\pi : C \to \{1, 2, \ldots, n\}$, for some $C \subset \{1, 2, \ldots, n\}$. The pathological situation when $C = \emptyset$ will be distinguished by using the term *empty partial permutation*. There is an obvious one to one correspondence between the set of permutations $\mathcal{S}_n$ and the set of the so called *permutation matrices* defined below.

Definition 1.5 *Let $X = (x_{ij})$ be an $n \times n$ matrix. If the entries x_{ij} fulfill the following conditions*

$$\sum_{i=1}^{n} x_{ij} = 1, \quad \textit{for} \quad 1 \le j \le n,$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \textit{for} \quad 1 \le i = \le n,$$

$$x_{ij} \in \{0, 1\} \quad \textit{for} \quad 1 \le i, j \le n,$$

then X is called a permutation matrix. The set of the $n \times n$ permutation matrices is denoted by Π_n.

The scalar *product* of two $n \times 1$ vectors $U = (u_i)$ and $V = (v_i)$ is denoted by $\langle U, V \rangle$ and defined by

$$\langle U, V \rangle = \sum_{i=1}^{n} u_i v_i$$

Consider three permutations $\phi, \psi, \rho \in \mathcal{S}_n$ such that $u_{\phi(1)} \le u_{\phi(2)} \le \ldots \le u_{\phi(n)}$, $u_{\rho(1)} \ge u_{\rho(2)} \ge \ldots \ge u_{\rho(n)}$ and $v_{\psi(1)} \ge v_{\psi(2)} \ge \ldots \ge v_{\psi(n)}$. As we will see in the

[2]It will always be clear from the context whether (i, j) is an edge in a graph or a transposition.

proposition concluding this subsection, the scalar products $\langle U^{\phi}, V^{\psi}\rangle$ and $\langle U^{\rho}, V^{\psi}\rangle$ play a special role. So, we find it convenient to use the following notations:

$$\langle U, V\rangle_- := \langle U^{\phi}, V^{\psi}\rangle \quad \text{and} \quad \langle U, V\rangle_+ := \langle U^{\rho}, V^{\psi}\rangle$$

Finally, we would like to recall an early result due to Hardy, Littlewood and Pòlya [87], which will turn out to be useful in many proofs through the rest of the thesis.

Proposition 1.1 *(Hardy, Littlewood and Pòlya [87], 1952)*
Let $U = (u_i)$ and $V = (v_i)$ be two $n \times 1$ vectors. Then, the following inequalities hold, for any $\pi \in \mathcal{S}_n$,

$$\langle U, V\rangle_- \leq \langle U^{\pi}, V\rangle \leq \langle U, V\rangle_+ \qquad \blacksquare$$

1.3.6 The problems QAP, GQAP, BiQAP and CAP

In this section we introduce the four main problems to be dealt with in this thesis. Moreover, we notice that all these problems fall into the class of combinatorial optimization problems according to Definition 1.3.

The quadratic assignment problem (QAP)

Given two $n \times n$ matrices, say A and B, an instance of the quadratic assignment problem with coefficient matrices A and B consists of finding a permutation $\pi_0 \in \mathcal{S}_n$ which minimizes the double sum

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij}$$

over all permutations $\pi \in \mathcal{S}_n$.

Now, let us give an equivalent formulation of QAP(A,B) following Definition 1.3. Namely, we specify a ground set $\mathcal{E}$, a set of feasible solutions $\mathcal{F}$ and a cost function f, such that instance of the combinatorial optimization problem with sum objective function defined as in Definition 1.3 with the specified $\mathcal{E}$, $\mathcal{F}$ and f, is equivalent to QAP(A,B). Through the rest of this paragraph we denote this instance by $\mathrm{CP}_{\mathrm{QAP(A,B)}}$. Define $\mathcal{E}$, $\mathcal{F}$ and f as follows:

$$\mathcal{E} := \{(i, j, k, l) \colon 1 \leq i, j, k, l \leq n\}$$

$$\mathcal{F} := \{\{(\pi(i), \pi(j), i, j) \colon 1 \leq i, j \leq n\} \colon \pi \in \mathcal{S}_n\}$$

$$f \colon \mathcal{E} \to \mathbb{R}$$
$$(i, j, k, l) \mapsto a_{ij} b_{kl}$$

Thus, the feasible solutions $X \in \mathcal{F}$ are of the form $\{(\pi(i)\pi(j), i, j)\colon 1 \le i, j \le n\}$ for some $\pi \in \mathcal{S}_n$. Obviously, there is a one to one correspondence between feasible solution $X \in \mathcal{F}$ and permutation $\pi \in \mathcal{S}_n$. Let us denote by X_π the solution corresponding to permutation π. Now $\mathrm{CP}_{\mathrm{QAP(A,B)}}$ looks as follows:

$$\min_{X \in \mathcal{F}} \sum_{x \in X} f(x)$$

or equivalently

$$\min_{\pi \in \mathcal{S}_n} \sum_{x \in X_\pi} f(x)$$

Since $\sum\limits_{x \in X_\pi} f(x) = \sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij}$, QAP(A,B) is equivalent to $\mathrm{CP}_{\mathrm{QAP(A,B)}}$. We do not make any difference between these two formulations of QAP(A,B). They will be used alternatively under the name QAP(A,B). We will also refer to both a permutation $\pi \in \mathcal{S}_n$ and a set X_π as feasible solutions of QAP(A,B).

A QAP(A,B) where at least one of the matrices A and B is symmetric is termed *symmetric QAP*. Sometimes, in the case that none of the matrices A and B is symmetric, QAP(A,B) is termed *asymmetric QAP*. A QAP(A,B) where A is a Euclidean or a grid matrix is termed *Euclidean QAP* or *grid QAP*, respectively.

The general quadratic assignment problem (GQAP)

Given two $n \times n$ matrices A, B and a subset $\mathcal{H}_n$ of the set $\mathcal{S}_n$ of permutations on $\{1, 2, \ldots, n\}$, an instance of the general quadratic assignment problem with coefficient matrices A and B and *permutations set* $\mathcal{H}_n$ consists of finding a permutation $\pi_0 \in \mathcal{H}_n$ which minimizes the double sum

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij}$$

over all permutations $\pi \in \mathcal{H}_n$. This instance is shortly denoted by $GQAP(A, B, \mathcal{H}_n)$. An obvious equivalent formulation of $GQAP(A, B, \mathcal{H}_n)$ in the scheme of Definition 1.3 is obtained by specifying the ground set $\mathcal{E}$ and the cost function f in the same way as for QAP(A,B) above, and the set of the feasible solutions as below:

$$\mathcal{F} := \{\{(\pi(i), \pi(j), i, j)\colon 1 \le i, j \le n\}\colon \pi \in \mathcal{H}_n\} \ .$$

Again, both formulation of the problem will be used alternatively under the name $GQAP(A, B, \mathcal{H}_n)$.

The biquadratic assignment problem (BiQAP)

Let two arrays $A = (a_{ijkl})$ and $B = (b_{ijkl})$ of n^4 elements each be given. An instance of the biquadratic assignment problems with coefficient arrays A and B consists of finding a permutation $\pi_0 \in \mathcal{S}_n$ which minimizes the multiple sum

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} a_{\pi(i)\pi(j)\pi(k)\pi(l)} b_{ijkl}$$

over all permutations $\pi \in \mathcal{S}_n$. This instance is denoted by BiQAP(A,B).

Similarly as in the case of QAP(A,B), an equivalent formulation of BiQAP(A,B) according to Definition 1.3 is trivially obtained by defining the ground set $\mathcal{E}$, the set of feasible solutions $\mathcal{F}$ and the cost function f as follows:

$$\mathcal{E} := \{(i,j,k,l,m,p,s,t) : 1 \le i,j,k,l,m,p,s,t \le n\}$$

$$\mathcal{F} := \{\{(\pi(i),\pi(j),\pi(k),\pi(l),i,j,k,l) : 1 \le i,j,k,l \le n\} : \pi \in \mathcal{S}_n\}$$

$$f : \mathcal{E} \to \mathrm{I\!R}$$
$$(i,j,k,l,m,p,s,t) \mapsto a_{ijkl} b_{mpst}$$

Both formulations will be used under the name BiQAP(A,B).

The communication assignment problem (CAP)

Let an $n \times n$ matrix $T = (t_{ij})$ and a graph $G = (V, E)$ be given, where $|V| = n$. Matrix T is assumed to be the *communication matrix* of a system of communicating centers C_i, $1 \le i \le n$, i.e., for all pairs (i,j), $1 \le i,j \le n$, t_{ij} is the rate of messages transmitted from center C_i to center C_j, per time unit. An *embedding* π of the centers C_i into the vertices of graph G is a one to one mapping of V to $\{C_1, C_2, \ldots, C_n\}$. When identifying each center C_i with its index i and assuming the vertices of V to be labeled by numbers $1, 2, \ldots, n$, an embedding is equivalently represented by a permutation of $\{1, 2, \ldots, n\}$. Thus, $\pi(i)$ is the index of the center located at vertex i by embedding π.

Let us denote by $conn(i,j)$ the arbitrarily ordered set of paths in G with startvertex i and endvertex j, for each ordered pair of vertices (i,j) in V:

$$conn(i,j) = \left\langle P_1^{(i,j)}, P_2^{(i,j)}, \ldots, P_{|conn(i,j)|}^{(i,j)} \right\rangle .$$

Consider now a fixed embedding $\pi \in \mathcal{S}_n$. A *routing pattern* ρ is a function which maps a pair (i,j) to a $|conn(i,j)| \times 1$ vector[3] $\rho(i,j) = (y_1^{(i,j)}, y_2^{(i,j)}, \ldots, y_{|conn(i,j)|}^{(i,j)})$, where

$$\begin{aligned} \sum_{k=1}^{|conn(i,j)|} y_k^{(i,j)} &= t_{\pi(i)\pi(j)} && (1.1) \\ y_k^{(i,j)} &\ge 0 && 1 \le k \le |conn(i,j)| \end{aligned}$$

[3]For the sake of simplicity, we use only one pair of parenthesis and write $\rho(i,j)$, instead of $\rho((i,j))$.

$y_l^{(i,j)}$ equals the amount of messages sent from $C_{\pi(i)}$ to $C_{\pi(j)}$ along path $P_l^{(i,j)}$, for $1 \le l \le |conn(i,j)|$. Thus, a routing pattern determines how to split the messages sent from $C_{\pi(i)}$ to $C_{\pi(j)}$ among the elements (paths) of $conn(i,j)$, for each ordered pair of vertices (i,j).

Equation (1.1) shows that the routing pattern ρ actually depends on the fixed embedding π. Namely, for each embedding π, there is a set of routing patterns associated with it, called *feasible routing patterns with respect to* π and defined by the following system of equalities:

$$\begin{array}{rcll} \sum\limits_{k=1}^{|conn(i,j)|} y_k^{(i,j)} & = & t_{\pi(i)\pi(j)} & \text{for all } 1 \le i,j \le n \\ & & & \\ y_l^{(i,j)} & \ge & 0 & \text{for } 1 \le i,j \le n \text{ and } 1 \le l \le |conn(i,j)| \end{array} \tag{1.2}$$

Each time time we consider a pair (π, ρ), where π is an embedding and ρ is a routing pattern, we implicitly assume ρ to be a feasible routing pattern with respect to embedding π.

Given an embedding π and a feasible routing pattern ρ, we define the *noise at vertex k of G* to be the overall amount of messages passing through center $C_{\pi(k)}$ as an intermediate center[4]. This amount of messages, which apparently depends on π and ρ, is denoted by $N_{\pi,\rho}(k)$. The goal is to find an embedding $\pi \in \mathcal{S}_n$ and a (feasible) routing pattern ρ such that $\max\{N_{\pi,\rho}(k) : 1 \le k \le n\}$ is minimized, i.e.,

$$\min_{\pi \in \mathcal{S}_n, \rho} \max_{1 \le k \le n} N_{\pi,\rho}(k).$$

This problem is called *the communication assignment problem* and the instance defined by matrix T and graph G is shortly denoted by CAP(T,G).

In the case that G is a tree, which is also the case we mostly deal with, there is a unique path joining each pair of vertices (i,j), i.e., $|conn(i,j)| = 1$. Therefore, for any fixed embedding π, there is only one feasible routing pattern, namely, the one which sents the whole amount of messages $t_{\pi(i)\pi(j)}$ through the path $P(i,j)$, the unique element of $conn(i,j)$. Therefore, the noise at vertex k depends only on the embedding π and is given by the following equality

$$N_\pi(k) = \sum_{(i,j): k \in P(i,j)} t_{\pi(i)\pi(j)}\,, \qquad 1 \le k \le n\,. \tag{1.3}$$

In the case that G is a tree, CAP(T,G) can be equivalently formulated to fit in the scheme of Definition 1.3 for a bottleneck problem. Denote by $\{1,2,\ldots,n\}^2$ the set of all ordered pairs (i,j) for $1 \le i,j \le n$. In the following we define a ground

[4]Here, the word "intermediate" means that the messages sent by center $C_{\pi(k)}$ itself to some other center or sent by some other center to $C_{\pi(k)}$ are not taken into account.

set $\mathcal{E}$, a set $\mathcal{F}$ of feasible solutions and a cost function f. The ground set consist of arbitrary sets of ordered pairs (i,j), with $1 \le i,j \le n$.

$$\mathcal{E} := 2^{\{1,2,\dots,n\}^2}$$

The feasible solutions $X \in F$ are of the form $\{\{(\pi(i),\pi(j)) \colon k \in P(i,j)\} \colon 1 \le k \le n\}$, for some $\pi \in \mathcal{S}_n$. Thus the set of feasible solutions is given as follows:

$$\mathcal{F} := \left\{ \left\{ \{(\pi(i),\pi(j)) \colon k \in P(i,j)\} \colon 1 \le k \le n \right\} \colon \pi \in \mathcal{S}_n \right\}$$

Obviously, there exists a one to one correspondence between the feasible solutions and the permutations in $\mathcal{S}_n$. Namely, the feasible solution corresponding to $\pi \in \mathcal{S}_n$, denoted by X_π, is given as

$$X_\pi = \left\{ \{(\pi(i),\pi(j)) \colon \text{for ordered pairs } (i,j) \text{ such that } k \in P(i,j)\} \colon 1 \le k \le n \right\}$$

The cost function $f \colon \mathcal{E} \to \mathbb{R}$ maps an element $x \in \mathcal{E}$ to the sum $\sum_{(ij)\in x} t_{ij}$. Clearly, according to this definition we have

$$\max_{x \in X_\pi} f(x) = \max_{1 \le k \le n} \sum_{(ij) \colon k \in P(i,j)} t_{\pi(i)\pi(j)} \qquad (1.4)$$

Now, the instance of the bottleneck problem defined by $\mathcal{E}$, $\mathcal{F}$ and F looks as follows:

$$\min_{\pi \in \mathcal{S}_n} \max_{x \in X_\pi} f(x)$$

Considering equalities (1.3) and (1.4), it is easy to see that the above instance of the bottleneck problem is equivalent to CAP(T,G), in the case that G is a tree.

Part I

The Quadratic Assignment Problem (QAP): A Brief Overview

Chapter 2

Getting Acquainted with the QAP

This chapter is a general introduction on the Quadratic Assignment Problem, shortly denoted by QAP. It gives a general idea about the problem itself and about the numerous methods which have been derived for dealing with it. This chapter should provide the reader with a clear understanding of the problem and the extent of its applications and difficulty. We try to explain the combinatorial nature of the QAP and shortly describe its place with respect to other combinatorial optimization problems. Moreover, this chapter gives a general overview of directions of research on QAP and shortly summarizes the achieved results.

2.1 Introduction

Problem Statement

Let a set $\mathcal{N} = \{1, 2, \ldots, n\}$ and two $n \times n$ matrices $A = (a_{ij})$, $B = (b_{ij})$ be given. The Quadratic Assignment Problem with coefficient matrices A and B, shortly denoted by QAP(A,B), can be stated as follows:

$$\min_{\pi \in \mathcal{S}_n} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij} \, , \tag{2.1}$$

where $\mathcal{S}_n$ is the set of permutations of $\{1, 2, \ldots, n\}$. That is, the QAP(A,B) is the problem of finding a permutation $\pi \in \mathcal{S}_n$ which minimizes the double sum in the above formulation. Obviously, the value of this sum depends on matrices A and B and on the permutation π. To formalize these dependencies we denote:

$$Z(A, B, \pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij}$$

Function $Z(A, B, \pi)$ is called *objective function* of the QAP(A,B) and a permutation π_0 which minimizes it over $\mathcal{S}_n$ is called *an optimal solution* to the QAP(A,B). The corresponding value of the objective function, $Z(A, B, \pi_0)$, is called *the optimal value* of the QAP(A,B). The term “quadratic” in the name of the problem is related to a formulation of the problem as an integer program with quadratic cost function.

In the literature, the QAP defined in (2.1) is often addressed to as the *Koopmans-Beckmann QAP* [105] in order to distinguish it from a more general formulation of the problem due to Lawler [115]:

$$\min_{\pi \in \mathcal{S}_n} \sum_{i=1}^{n} \sum_{j=1}^{n} d_{\pi(i)\pi(j)ij} \,, \tag{2.2}$$

where d_{ijkl} are given reals, for $i, j, k, l = 1, 2, \ldots, n$. Obviously, in the case that the numbers d_{ijkl} fulfill the equalities $d_{ijkl} = a_{ij}b_{kl}$, for $1 \le i, j, k, l \le n$, the problem given in (2.2) is equivalent to the QAP(A,B) defined in (2.1). In this thesis we consider only the QAP(A,B) as defined in (2.1) and not the more general version introduced by Lawler. Anyway some of the results presented in the first part extend also to the latter problem.

There is another slightly different problem addressed to as QAP, used and investigated by several authors. The difference concerns the objective function which has an additional linear term. Namely, the problem input consists not only of two matrices A, B but also of an additional matrix $C = (c_{ij})$, which contains the coefficients of a linear part of the objective function:

$$\min_{\pi \in \mathcal{S}_n} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij} + \sum_{i=1}^{n} c_{\pi(i)i} \right) \quad \textit{or}, \tag{2.3}$$

$$\min_{\pi \in \mathcal{S}_n} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} d_{\pi(i)\pi(j)ij} + \sum_{i=1}^{n} c_{\pi(i)i} \right) \tag{2.4}$$

In the case that $c_{ij} = 0$, for all $1 \le i, j \le n$, we get the problems formulated in (2.1) and (2.2), respectively. Again some of the results presented in the first part apply also to this last problem.

Applications

Historically, the first occurrence of QAP dates back to 1957, when Koopmans and Beckmann [105] derived the QAP as a mathematical model of assigning a set of economic activities to a set of locations. Thus, the QAP occurred at first in the context of facility location problems, which still remain one of its major applications. Other major applications of QAP are related to so called *wiring* problems.

Facility location. In this context n facilities are to be assigned to n locations. $A = (a_{ij})$ is the flow matrix, i.e. a_{ij} is the flow of materials moving from facility i to facility j, per time unit, and $B = (b_{ij})$ is the distance matrix, i.e., b_{ij} represents the distance from location i to location j. The cost of simultaneously locating facility $\pi(i)$ to location i and facility $\pi(j)$ to location j is $a_{\pi(i)\pi(j)}b_{ij}$. Obviously, an assignment of all facilities to locations can mathematically be represented by a permutation $\pi \in \mathcal{S}_n$. In this model, the total cost of an assignment π of all facilities to locations is equal to $Z(A, B, \pi)$. The objective is to find an assignment π of locations to facilities such that the total cost $Z(A, B, \pi)$ is minimized. This amounts to solving QAP(A,B). Concrete applications of QAP in a plant location context are described by Dickey and Hopkins in [49] and by Elshafei [53]. In [49] a campus planning model is presented, whereas in [53] the design of a hospital layout is modeled as a QAP.

In the following we will often refer to QAP in the facility-location context. Thus, the terms "facility" and "location" will be often abstractly used, even if there is no concrete occurrence of a facility location problem.

Wiring problems. In this group of problems a number of modules have to be placed on a board. The modules are pairwise connected by a number of wires. We wish to find a placement of the modules on the board, such that the total length of the needed wires is minimized. Obviously, this problem can be modeled as a QAP. Steinberg [171] describes in details an application of QAP in backboard wiring in electronics.

Other applications. Many other proposed applications of QAP arise in scheduling [71], in parallel and distributed computing [17], in statistical data analysis [35, 93, 178], in the design of control panels and typewriter keyboards [33, 144], in sports [89], in chemistry [179], in archeology [107] and in balancing turbine runners [112, 167]. Some of these applications are considered in details in the second part of this thesis. Recently, Malucelli [124] proposes some applications of the QAP and its relatives in the field of transportation.

Organization of the chapter

In the second section of this chapter we give alternative equivalent formulations of the QAP which are due to several authors. These formulations, derived when considering the problem from different points of view, have led to different approaches to deal with the QAP. In Section 2.3 computational complexity aspects of QAPs are discussed. Section 2.4 reviews the most important results on the computation of lower bounds for QAPs and also presents some of the most significant results on incorporating these lower bounds in branch and bound algorithms. Different heuristic approaches for QAPs are reviewed in Section 2.5. Section 2.6 provides some results on generating instances of QAPs with known optimal solution. The significance of such results becomes noticeable for testing heuristics with respect to the "quality" of their suboptimal solutions. Finally, in Section 2.7 we consider the interesting asymptotic behavior of QAPs in the context of a whole group of combinatorial optimization problem which share a similar behavior.

2.2 Different formulations of the QAP and linearizations

We usually refer to QAP(A,B) as defined by Definition (2.1). The main reason for this is that the problem formulation in (2.1) expresses the QAP's combinatorial structure better than the alternative equivalent formulations. However, different formulations of the QAP are helpful, when working in different areas of discrete optimization.

2.2.1 Two alternative formulations for QAPs

Koopmans and Beckmann formulation. Let us first introduce the formulation used initially by Koopmans and Beckmann [105], as they introduced the problem. Their formulation basically relies on the one to one correspondence between the set of permutations $\mathcal{S}_n$ and the set of *permutation matrices* as defined in Subsection 1.3.5. Using permutation matrices it is easy to see that QAP(A,B) is equivalent to the following minimization problem on the set of permutation matrices:

$$\begin{array}{lll} \min & \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} a_{ij}b_{kl}x_{ik}x_{jl} & \\ \text{subject to} & & \\ & \sum_{i=1}^{n} x_{ij} = 1 & 1 \le j \le n \\ & \sum_{j=1}^{n} x_{ij} = 1 & 1 \le i \le n \\ & x_{ij} \in \{0,1\} & 1 \le i,j \le n \end{array} \tag{2.5}$$

The problem formulation given in (2.5) is called *Koopmans-Beckmann formulation* of QAP.

The equivalence of (2.1) and (2.5) becomes clear when considering QAP(A,B) in the facility location context. Namely, $x_{ij} = 1$ if facility i is placed at location j and $x_{ij} = 0$ otherwise. The other restrictions in (2.5) formalize the facts that a facility is assigned to exactly one location and that to each location is assigned exactly one facility. A term $a_{ij}b_{kl}x_{ik}x_{jl}$ contributes to the objective function with a value equal to $a_{ij}b_{kl}$ if and only if $x_{ik} = x_{jl} = 1$, that is, iff facility i is assigned to location k and facility j is assigned to location l simultaneously.

Trace formulation. Using the permutation matrices as defined above another equivalent formulation of QAP can be derived. Namely, for a QAP instance QAP(A,B) of size n, a function $f_{A,B}$ can be defined on the set Π_n of permutation matrices:

$$f_{A,B}\colon \Pi_n \to \mathbb{R}$$

$$X \mapsto \operatorname{tr}(AXBX^t),$$

where the superscript ^{t} denotes the transposed of the corresponding matrix and $tr(A)$ is the trace of matrix A, i.e. $\mathrm{tr}(A) = \sum_{i=1}^n a_{ii}$, for any $n \times n$ matrix A. Now, QAP(A,B) is equivalent to the following minimization problem on the set of permutation matrices Π_n:

$$\min_{x \in \Pi_n} f_{A,B}(X) = \mathrm{tr}(AXBX^t) \tag{2.6}$$

This formulation was introduced by Edwards in [51, 52]. It may be used for a flexible algebraic manipulation of the problem data. In Section 2.4.2 we will see how this formulation of the problem is used for deriving the so called *eigenvalue related lower bounds*.

2.2.2 Linearizations

When dealing with QAPs, it seems that the quadratic form in its objective function hopelessly destroys every attempt to efficiently solve the problem. One of the first ideas to avoid this inconvenient feature of the problem was probably the so called *linearization of the QAP*. That is, getting rid of the quadratic form in the QAP objective function by finding an equivalent linear formulation of it. A more or less complete list of references to various QAP linearizations can be found in [142]. Moreover, some of them are described in detail in [22]. Here, we present only two of the numerous linearizations of QAP: the linearization proposed by Kaufmann and Broeckx [102] and the linearization proposed by Frieze and Yadegar [63].

Kaufman and Broeckx linearization. Kaufman and Broeckx [102] derived an equivalent formulation of the QAP as a mixed integer program with $O(n^2)$ binary variables, $O(n^2)$ real variables and $O(n^2)$ constraints. This linearization, which is probably the smallest one with respect to the number of variables and constraints, works also for the more general QAP defined in (2.2).

Consider the Koopmans and Beckmann formulation (2.5) of QAP(A,B). For $1 \le i, j \le n$, let us introduce real variables w_{ij} by

$$w_{ik} := x_{ik} \sum_{j=1}^n \sum_{l=1}^n a_{ij} b_{kl} x_{jl}$$

Using these new variables the objective function of QAP(A,B) in (2.5) can be linearized:

$$\sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \sum_{l=1}^n a_{ij} b_{kl} x_{ik} x_{jl} = \sum_{i=1}^n \sum_{k=1}^n w_{ik} \tag{2.7}$$

Moreover, define new constants d_{ik}, $1 \le i, k \le n$, by

$$d_{ik} = \sum_{j=1}^n \sum_{l=1}^n a_{ij} b_{kl} \tag{2.8}$$

The following theorem due to Kaufman and Broeckx, whose proof can be found also in [22], gives a linearization of QAP(A,B).

Theorem 2.1 (Kaufman and Broeckx [102], 1978)
QAP(A,B) given in (2.1) is equivalent to the following mixed integer linear program with n^2 binary variables, n^2 real variables and $n^2 + 2n$ restrictions

$$\begin{array}{ll} \textit{minimize} & \sum_{i=1}^n \sum_{k=1}^n w_{ik} \\ \textit{subject to} & \\ & \sum\limits_{i=1}^n x_{ik} = 1 \qquad 1 \le k \le n \\ & \sum\limits_{k=1}^n x_{ik} = 1 \qquad 1 \le i \le n \\ & d_{ik}x_{ik} + \sum_{j=1}^n \sum_{l=1}^n a_{ij}b_{kl}x_{jl} - w_{ik} \ge d_{ik} \qquad 1 \le i,k \le n \\ & x_{ik} \in \{0,1\}\,,\ w_{ik} \ge 0\,, \qquad 1 \le i,k \le n \end{array} \tag{2.9}$$

where d_{ik} are defined in (2.8). ■

Notice that all QAP linearizations are extremely large, i.e., the number of their constraints and variables is very high. Under these conditions, even powerful tools to cope with linear integer programs such as Benders' decomposition [13] or cutting planes [14] do not help a lot. It turns out that for relatively small QAPs even solving the relaxed linear program is computationally a hard job.

Frieze and Yadegar linearization. Again, consider the Koopmans and Beckmann formulation of QAP(A,B) of size n in (2.5). Introduce n^4 new binary variables variables y_{ijkl} by:

$$y_{ijkl} := x_{ik}x_{jl} \quad \text{for} \quad 1 \le i,j,k,l \le n$$

By using these variables Frieze and Yadegar derive a linearization of QAP as shown by the following theorem.

Theorem 2.2 (Frieze and Yadegar [63], 1983)
QAP(A,B) in (2.5) is equivalent to the following mixed integer program with n^4 real variables, n^2 binary variables and $n^4 + 4n^3 + n^2 + 2n$ constraints:

$$\begin{array}{ll} \textit{minimize} & \sum\limits_{i=1}^n \sum\limits_{j=1}^n \sum\limits_{k=1}^n \sum\limits_{l=1}^n a_{ij}b_{kl}y_{ijkl} \\ \textit{subject to} & \\ & \sum\limits_{i=1}^n x_{ik} = 1 \qquad 1 \le k \le n \\ & \sum\limits_{k=1}^n x_{ik} = 1 \qquad 1 \le i \le n \\ & \sum\limits_{i=1}^n y_{ijkl} = x_{jl} \quad 1 \le j,k,l \le n \\ & \sum\limits_{j=1}^n y_{ijkl} = x_{ik} \quad 1 \le i,k,l \le n \\ & \sum\limits_{k=1}^n y_{ijkl} = x_{jl} \quad 1 \le i,j,l \le n \\ & \sum\limits_{l=1}^n y_{ijkl} = x_{ik} \quad 1 \le i,j,k \le n \end{array} \tag{2.10}$$

$$\begin{array}{ll} y_{iikk} = x_{ik} & 1 \le i,k \le n \\ x_{ik} \in \{0,1\} & 1 \le i,k \le n \\ 0 \le y_{ijkl} \le 1 & 1 \le i,j,k,l \le n \end{array} \qquad \blacksquare$$

Frieze and Yadegar use this mixed integer program formulation to derive lower bounds for the QAP by using Lagrangean relaxation. As reported in [124], Queyranne observes that only half of the variables y_{ijkl} are really needed in (2.10), since $y_{ijkl} = y_{jilk}$, for all $1 \le i,j,k,l \le n$. Considering this formulation of the QAP, he proposes some valid inequalities for the QAP polytope. They have not been proved to be facet defining. Malucelli describes in [124] how to use these inequalities for improving the performance of lower bound computations when solving relaxations of linear programming QAP formulations. Let us notice that the investigation of integer programming formulations of QAP might be helpful towards a facial characterization of the QAP polytope, which has not been proposed up to now.

2.3 Computational complexity aspects of QAPs

This section is dedicated to computational complexity aspects of QAPs. All results in this section will bring evidence to the fact that the QAP is a "very hard" problem, at least from the theoretical point of view. Other results presented in the coming sections of this chapter will show that the complexity of this problem unfortunately extends also to its practical aspects.

We first show that QAP belongs to the class of $\mathcal{NP}$-hard problems. Then we discuss the complexity of finding an approximate solution to QAPs in general and to QAPs with an arbitrarily large optimal value in particular, showing that this is also a hard problem. In the second part of this section the problem of finding a locally optimal solution to the QAP is considered. Two underlying neighborhood structures which lead to PLS-complete problems are described.

2.3.1 The complexity of optimally and approximately solving QAPs

The following two theorems reformulate two early results obtained by Sahni and Gonzalez [163] in 1976. Trying to restrict ourselves on using only the basic concepts from the complexity theory introduced in the previous subsection, our formulations slightly differ from those given in [163].

Theorem 2.3 (Sahni and Gonzalez [163], 1976)
The quadratic assignment problem is strongly NP-hard.

Proof. The proof consists of showing that the existence of a polynomial algorithm for solving QAPs with entries of coefficient matrices being equal to 0, 1 or 2 leads to a polynomial algorithm for an NP-complete decision problem. By definition, this implies that the QAP is strongly NP-hard. The above mentioned decision problem is the Hamiltonian cycle problem (for short HC), which is well known to be NP-complete (see [100, 68]).

(HC) Given a graph $G = (V, E)$ with vertex set V and edge set E. Does G contain a Hamiltonian cycle?

Assume that there exists a polynomial time algorithm for solving QAPs with coefficients equal to 0, 1 or 2. Consider an arbitrary instance of the Hamiltonian cycle problem, i.e., an arbitrary graph $G = (V, E)$. Let $|V| = n$, $V = \{v_1, v_2, \ldots, v_n\}$. Define two $n \times n$ matrices $A = (a_{ij})$ and $B = (b_{ij})$ as follows:

$$a_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 2 & \text{otherwise} \end{cases} \quad b_{ij} = \begin{cases} 1 & \text{if } j = i+1, 1 \le i \le n-1 \text{ or } i = n, j = 1 \\ 0 & \text{otherwise} \end{cases},$$

and consider QAP(A,B). It is very easy to see that the optimal value of QAP(A,B) is equal to n if and only if graph G contains a Hamiltonian cycle. Thus, we just translate the given instance of HC into an instance of QAP as above and apply to the QAP instance the polynomial algorithm which is assumed to exist. Then, check whether the solution provided by this algorithm is equal to n. All this can be done in polynomial time with respect to the size of the HC instance. Therefore, we would have a polynomial algorithm for the Hamiltonian cycle problem. ■

In addition, Sahni and Gonzalez have proven that even finding an ϵ-approximate solution for QAPs is a hard problem, in the sense that the existence of a polynomial ϵ-approximate algorithm implies $\mathcal{P} = \mathcal{NP}$.

Let us first introduce the notion of an ϵ-approximate algorithm for the QAP (or, similarly, for any minimization problem in general).

Definition 2.1 *Given a real number $\epsilon > 0$, an algorithm Υ for the QAP is said to be an ϵ-approximate algorithm if and only if for every instance QAP(A,B) the following holds:*

$$\left| \frac{Z(A, B, \pi_\Upsilon) - Z(A, B, \pi_{opt})}{Z(A, B, \pi_{opt})} \right| \le \epsilon\,, \tag{2.11}$$

where π_Υ is the solution to QAP(A,B) computed by algorithm Υ and π_{opt} is an optimal solution to QAP(A,B). The solution of QAP(A,B) computed by an ϵ-approximate algorithm is called an ϵ-approximate solution.

Theorem 2.4 (Sahni and Gonzalez [163], 1976)
For an arbitrary $\epsilon > 0$, the existence of a polynomial ϵ-approximate algorithm for QAP implies $\mathcal{P} = \mathcal{NP}$.

Proof. The proof consists on showing that the existence of a polynomial ϵ-approximate algorithm for QAP implies the existence of a polynomial algorithm for the Hamiltonian cycle problem, which is an NP-complete decision problem.

Indeed, assume that Υ is a polynomial ϵ-approximate algorithm for QAP. We show how to design a polynomial algorithm for HC making use of algorithm Υ. Consider an instance of HC, i.e., an arbitrary graph $G = (V, E)$ with vertex set $V = \{v_1, \ldots, v_n\}$. Construct (in polynomial time) a QAP instance QAP(A,B) by defining two $n \times n$ coefficient matrices $A = (a_{ij})$ and $B = (b_{ij})$ as follows:

$$a_{ij} = \begin{cases} 1 & \text{if} \quad (v_i, v_j) \in E \\ \omega & \qquad \text{otherwise} \end{cases}$$

$$b_{ij} = \begin{cases} 1 & \text{if} \quad j = i+1, 1 \leq i \leq n-1 \text{ or } i = n, j = 1 \\ 0 & \qquad \text{otherwise} \end{cases} ,$$

where $\omega > 1 + n\epsilon$. Apply algorithm Υ to QAP(A,B). Denote the solution provided by Υ and an optimal solution to QAP(A,B) by π_Υ and π_{opt} respectively. Inequality (2.11) implies

$$Z(A, B, \pi_{\text{opt}}) \geq \frac{Z(A, B, \pi_\Upsilon)}{1 + \epsilon}$$

The last inequality shows that if $Z(A, B, \pi_\Upsilon) > n(1 + \epsilon)$, then $Z(A, B, \pi_{\text{opt}}) > n$, and therefore G does not contain a Hamiltonian cycle. Viceversa, if G does not contain a Hamiltonian cycle, then $Z(A, B, \pi) > n - 1 + \omega = n(1 + \epsilon)$, for all $\pi \in \mathcal{S}_n$. Thus, $Z(A, b, \pi_\Upsilon) > n(1 + \epsilon)$. Hence, HC in G has an answer "yes" if and only if $Z(A, B, \pi_\Upsilon) > n(1 + \epsilon)$. Our polynomial algorithm for solving HC consists of three steps. First, translate the HC instance into an appropriate QAP instance as above. Second, apply the algorithm Υ to the QAP instance. Third, check the objective function value $Z(A, B, \pi_\Upsilon)$ of the solution π_Υ provided by algorithm Υ. Noticing that these three steps can be performed in polynomial time completes the proof. ■

Considering the general belief that $\mathcal{P} \neq \mathcal{NP}$, Theorem 2.4 shows that it is very unlikely to find a polynomial ϵ-approximate algorithm for QAP, for any $\epsilon > 0$. Thus, solving QAP to optimality or even finding an ϵ-approximate solution to it are considered to be hard problems. Queyranne [146] derives a stronger result which further confirms the widely spread belief on the inherent difficulty of QAP. Before formulating Queyranne's result we need one more definition related to the theoretical analysis of the performance of heuristics (cf. [68]).

Definition 2.2 *Let Υ be a heuristic for the QAP. The performance ratio $R_\Upsilon(A, B)$ for heuristic Υ with respect to an instance QAP(A,B) is given as*

$$R_\Upsilon(A, B) = \frac{Z(A, B, \pi_\Upsilon)}{Z(A, B, \pi_{opt})} ,$$

where π_Υ is the solution produced by Υ when applied to QAP(A,B) and π_{opt} is an optimal solution to QAP(A,B).
The absolute performance ratio R_Υ for heuristic Υ is given as

$$R_\Upsilon = \inf\{r \geq 1 \colon R_\Upsilon(A, B) \leq r \ \textit{ for all matrices } A, \ B\}$$

The asymptotic performance ratio R_{Υ}^{∞} for heuristic Υ is given by the following equality:

$$R_{\Upsilon}^{\infty} = \inf\{r \geq 1 \colon \exists m \in \mathbb{N},\ \ R_{\Upsilon}(A,B) \leq r \textit{ for all } A,\ B \textit{ with } Z(A,B,\pi_{opt}) \geq m\}$$

Theorem 2.5 (Queyranne [146]), 1988)
The existence of a polynomial heuristic Υ for the Euclidean QAP with a bounded asymptotic performance ratio, that is the existence of a constant $K \geq 1$ such that $R_{\Upsilon}^{\infty} \leq K$, implies $\mathcal{P} = \mathcal{NP}$. ■

In other words, the last theorem states that unless $\mathcal{P} = \mathcal{NP}$, for each polynomial heuristic Υ and for each $K_1, K_2 \in \mathbb{N}$, there exist a Euclidean matrix A and an arbitrary matrix B, such that the optimal value of QAP(A,B) is least K_1, $Z(A,B,\pi_{opt}) \geq K_1$, and

$$\frac{Z(A,B,\pi_{\Upsilon})}{Z(A,B,\pi_{opt})} \geq K_2\ ,$$

where π_{Υ} is the permutation produced by Υ when applied to QAP(A,B).

2.3.2 Local search complexity for QAPs

In this subsection we consider the intriguing question "Is it easy to find a locally optimal solution for the QAP?"

Obviously the answer depends on the definition of the neighborhood structure. Murthy, Pardalos and Li [134] introduce a neighborhood structure for QAPs and show the corresponding local search problem is PLS-complete. The proposed neighborhood structure is similar to the neighborhood structure proposed by Kernighan and Lin [103] for the graph partitioning problem. For this reason we will call it a *K-L type neighborhood structure for the QAP*

A K-L neighborhood structure for the QAP. Consider QAP(A,B) and a permutation $\pi_0 \in \mathcal{S}_n$. A *swap* of permutation π_0 is a permutation obtained by applying a transposition (i,j) to π_0. That is, in the facility location context a swap is obtained by interchanging the facilities assigned to two locations i and j. A *greedy swap* of permutation π_0 is a swap π_1 which minimizes the difference $Z(A,B,\pi) - Z(A,B,\pi_0)$ over all swaps π of π_0. Let $\pi_0, \pi_1, \ldots, \pi_l$ be a set of permutations in $\mathcal{S}_n$, each of them being a greedy swap of the preceding one. Such a sequence is called *monotone* if for each pair of permutations π_i, π_j $\{i_1, i_2\} \cap \{j_1, j_2\} = \emptyset$, where π_1, π_2 are obtained by applying transpositions (i_1, j_1), (i_2, j_2) to the preceding permutations, respectively. The *neighborhood of permutation* π_0 consists of all permutations which occur in the (unique) maximal monotone sequence of greedy swaps starting with permutation π_0. So, we have defined a neighborhood structure for QAP. Let us denote it by $\mathcal{N}_{K-L}$. Given a QAP(A,B) of size n and a permutation $\pi \in \mathcal{S}_n$, the cardinality of $\mathcal{N}_{K-L}(\pi)$ is equal to $\lceil n/2 \rceil$.

It is easily seen that the local search problem $(\text{QAP}, \mathcal{N}_{\text{K-L}})$ is a member of $\mathcal{PLS}$. Indeed, generating a feasible solution to the QAP and computing the corresponding objective function value can obviously be done in polynomial time. Moreover, we can determine in polynomial time whether a permutation π is optimal and if it is not, generate a better permutation among its $\lceil n/2 \rceil$ neighbors. In [142] it is shown that a PLS-complete search problem, namely, the graph partitioning problem with the neighborhood structure defined by Kernighan and Li [103] is $\mathcal{PLS}$-reducible to $(\text{QAP}, \mathcal{N}_{\text{K-L}})$. Thus the following theorem holds

Theorem 2.6 (Murthy, Pardalos and Li [142], 1994)
The local search problem $(\text{QAP}, \mathcal{N}_{\text{K-L}})$*, where* $\mathcal{N}_{K-L}$ *is the Kernighan-Lin type neighborhood structure for QAP, is* $\mathcal{PLS}$*-complete.* ■

In [134] it is reported that the generic local search algorithm presented in Figure 1.1 in Subsection 1.3.2 performs very well when the proposed Kernighan-Lin type neighborhood structure is used. As a consequence of the PLS-completeness, it can be shown that this good local search algorithm has exponential worst case time complexity.

It can be also shown that the QAP with an even much simpler neighborhood structure is in PLS-complete. Here, the so called pair-exchange (or 2-opt) neighborhood structure is meant. The pair-exchange neighborhood of a permutation $\pi \in \mathcal{S}_n$ consists of all permutations in $\mathcal{S}_n$ obtained by applying some transposition (i, j) to π. Let us denote this neighborhood structure by $\mathcal{N}_2$. Thus, $\mathcal{N}_2(\pi) = \{(i, j) \circ \pi : i \neq j,\ 1 \leq i, j \leq n\}$.
In [166] it is proven that the graph partitioning problem mentioned above with a neighborhood structure analogous to $\mathcal{N}_2$ is PLS-complete. Using the same $\mathcal{PLS}$-reduction as in [134] the following result can be proven immediately:

Theorem 2.7 *The local search problem* $(\text{QAP}, \mathcal{N}_2)$*, where* $\mathcal{N}_2$ *is the pair exchange neighborhood structure for QAP, is a PLS-complete problem.* ■

Again, from the $\mathcal{PLS}$-completeness follows that the generic local search algorithm presented in Figure 1.1 has exponential worst case time complexity.

At this point, we should mention that there are no known local criteria to decide how good a locally optimal solution is with respect to a global one. From the complexity point of view, deciding whether a given permutation is an optimal solution to a given instance of QAP, is a hard problem.

Theorem 2.8 (Papadimitriou and Wolfe [140], 1985)
Let Υ *be an algorithm which outputs whether an input permutation* $\pi \in \mathcal{S}_n$ *is a (globally) optimal solution to a QAP(A,B) of size n, where A and B are part of the input. If* Υ *is polynomial, then* $\mathcal{P} = \mathcal{NP}$*.* ■

2.4 Exact algorithms and lower bounds

This section gives a short summary on approaches and methods used for solving QAPs to optimality. Since the QAP is an NP-hard problem only implicit enumeration methods are known for solving it to optimality. Branch and bound algorithms seem to be the most successful implicit enumeration approaches to QAPs. Anyway, even the results achieved by applying the best existing exact algorithms are modest: problems of size larger than 20 cannot be solved in reasonable time. Even problems of size larger than 15 are generally considered difficult. The low efficiency of branch and bound algorithms is partially due to a lack of efficient bounding approaches for problems of relatively large size, which again indirectly confirms the inherent difficulty of the QAP. As the performance of branch and bound methods depends very much on the quality of the involved lower bounds, quite a lot of efforts are spent for designing efficient lower bounding schemes.

In this section, branch and bound approaches are focused among different methods used for solving QAPs to optimality. As lower bounds are widely accepted to be the most decisive component for the performance of branch and bound methods, we go a little bit into details when discussing different approaches for the lower bound calculation. On the other side, we only review the principal fundamental ideas in the design of branch and bound algorithms for QAPs.

2.4.1 Solving QAPs to optimality

There are essentially three types of algorithms which have been used for the implicit enumeration:

1. cutting plane methods
2. dynamic programming
3. branch and bound approaches

Cutting plane methods. The first to use cutting plane type methods for QAPs were Bazaraa and Sherali [13, 14]. As these methods use (mixed) integer linear programming formulations whose size is extremely large, even for problems of moderate size, the computational experience with these methods is not satisfactory. Sometimes, even the computational bargain for solving the corresponding LP relaxations is heavy. On the other side, a facial characterization of the QAP polytope, which could make these solution methods more efficient does not exist. However, heuristics derived from cutting plane approaches produce good suboptimal solutions in early stages of the search. Consider, for example, the cutting plane type heuristics proposed in [23, 14].

Dynamic programming. In 1989, Christofides and Benavent [40] used a dynamic programming approach to solve a special case of QAP(A,B), the so called *tree-QAP*, where matrix B is the weighted adjacency matrix of a tree. Even this special case of the problem is NP-hard as shown in [41]. The method proposed in [40] is able to solve QAP instances of size up to 25 in reasonable time, whereas general QAPs of size larger than 20 are nowadays still considered intractable. Actually, the branch and bound scheme derived by Christofides and Benavent is very convenient as it produces quite small branch and bound trees. This is due to the very good quality of the involved lower bounds. These bounds are obtained by solving the Lagrangean relaxation of an integer programming formulation for tree-QAPs. The Lagrangean relaxations are solved by a fast dynamic programming approach.

Branch and Bound approaches. Since the QAP has been introduced, many authors have proposed sequential and parallel branch and bound approaches for this problem. These approaches can be classified in three main groups with respect to the branching rule:

1. Single assignment methods
2. Pair assignment methods
3. Relative positioning methods.

All these algorithms start with an *empty partial permutation* (i.e., in the facility location context, no facility is assigned to any location) and step by step extend it to a full permutation (a permutation which assigns all facilities to locations).

The single assignment methods were the first approaches applied to QAPs. Algorithms belonging to this group assign a single (not yet located) facility, say i, to a (not yet occupied) location, say j, at each branching step. From a historical point of view, single assignment methods date back to the algorithm proposed by Gilmore [72] for the QAP(A,B). This algorithm was then generalized by Lawler [115] for QAPs as in (2.2). Other single assignment branch and bound algorithms for QAPs have been proposed by several authors [29, 52, 96, 141, 126]. The choice of the above mentioned pair of indices (i, j) usually depends on the used bounding scheme. As almost all bounding schemes end up by solving a linear assignment problem the choice of the pair (i, j) is often based on the so called *alternative costs* p_{ij} given as

$$p_{ij} = \min\{\bar{c}_{ik}: k \neq j,\ 1 \leq k \leq n\} + \min\{\bar{c}_{kj}: k \neq i,\ 1 \leq k \leq n\}\ ,$$

where $\bar{c}_{ij}$ are the reduced cost of the last linear assignment problem solved when calculating the current lower bound. The alternative cost p_{ij} is a lower bound for the increment of the current lower bound, in the case that the assignment of i to j is fixed. Burkard proposed in [22] the maximization of the alternative costs as a natural criterion for choosing (i, j). Another rule to choose the pair (i, j) has been considered by Bazaraa and Kirca [12]. They use a combination of the following two criteria:

- Try to minimize the sum of all lower bounds at the next branch and bound tree level.
- Try to minimize the number of branches at the next branch and bound tree level.

Another interesting criterion for the choice of the pair (i,j) is proposed by Mautor and Roucairol in [126]. They also calculate the alternative costs p_{ij} as above. Assume that $\underline{z}$ and $\bar{z}$ are the lower and the upper bound at the current node of the branch and bound tree, respectively. If $p_{ij} + \underline{z} \geq \bar{z}$ then pair (i,j) is forbidden. Then i is chosen to be the index with the largest number of forbidden pairs having it as first element. The chosen index j is an allowed index, i.e., the pair (i,j) is not a forbidden one, with maximum sum of elements of the corresponding column of matrix (p_{ij}). Ties are broken by maximizing the alternative cost. This method seems to work pretty well and seems to produce a much smaller branch and bound tree than other methods.
The pair (i,j) could also be selected with respect to a previously fixed order, as proposed for example in [12]. The advantageous consequence of using this rule is that the partial permutation is completely determined by the position of the corresponding node in the branch and bound tree.

The pair assignment methods allocate a pair of facilities at a pair of locations at each branching step. Computational experiments developed by a number of authors [69, 111, 136] have shown that these algorithms do not produce good results.

Relative positioning methods. This type of branch and bound method was proposed by Mirchandani and Obata [130]. Here, the levels of the branch and bound tree do not correspond to the assignment of facilities to locations. The partial permutations at each level are determined in terms of distances between facilities, i.e., their relative positions. This approach is claimed to be appropriate for QAPs with sparse matrices.
Another interesting branching rule, which does not belong to any of the above groups, was developed by Roucairol [158]. It is called *polytomic* or *k-partite branching rule*. The branch and bound tree produced by this algorithm is not a binary tree as in most of the other methods. In this case, the solution of the last linear assignment problem solved for calculating the lower bound at the current note of the branch and bound tree is considered. Assume that this solution is the permutation $\rho \in \mathcal{S}_n$ (for a problem of size n). Let $\mathcal{S}_n^{(i)}$ be the subset of $\mathcal{S}_n$ consisting of those permutations $\pi \in \mathcal{S}_n$, such that $\pi(i) = \rho(i)$. Analogously, $\bar{\mathcal{S}}_n^{(i)}$ is the set of permutations $\pi \in \mathcal{S}_n$, such that $\pi(i) \neq \rho(i)$. Then, the current node is branched into $n+1$ new nodes, whose set of feasible solutions are given as follows $\mathcal{S}_n^{(1)}, \mathcal{S}_n^{(1)} \cap \bar{\mathcal{S}}_n^{(2)}, \ldots, \mathcal{S}_n^{(1)} \cap \mathcal{S}_n^{(2)} \cap \ldots \cap \mathcal{S}_n^{(n-1)} \cap \bar{\mathcal{S}}_n^{(n)}, \mathcal{S}_n^{(1)} \cap \mathcal{S}_n^{(2)} \cap \ldots \cap \mathcal{S}_n^{(n)}$. (Clearly the nodes corresponding to the two last feasible set are trivial in the sense that the first of these sets is empty and the second one consists only of permutation ρ.)

Concluding this paragraph, let us remark that better results on solving large size problems have been recently achieved by using parallel implementations [113, 141, 158, 44]. Probably, the progress in solving QAP instances of size $n = 20$ with state of the art algorithms is mostly due to hardware improvement. However, recent results confirm the effectiveness of combining the best available algorithmic ideas with the most suitable hardware.

Table 2.1 list some of the most celebrated branch and bound algorithms for QAPs, including the size of the problems they can solve.

Table 2.1: The evolution of branch and bound approaches to QAP

Authors	Year	Size
Gilmore	1962	8
Lawler	1963	8
Gavett and Plyter	1965	8
Burkard	1973	8
Bazaraa and Sherali	1980	6
Burkard and Derigs	1980	15
Bazaraa and Kirca	1983	15
Roucairol	1987	12
Pardalos and Crouse	1988	15
Mautor and Roucairol	1992	19
Clausen and Perregård	1994	20

Transformation of QAPs to optimization problems in graphs. As another effort to cope with the QAP some attempts to transform QAPs in well-studied graph theoretical problems can be mentioned. However, these approaches are not really proven to be successful in practice.
Malucelli [124] describes two attempts to transform the QAP into well known graph problems. Then he applies to these transformations the tools which have been elaborated for the graph problems. Namely, in [124] transformations of the QAP to *max clique problems* and *max cut problems* with additional constraints are proposed.
In the first approach, the max clique formulation of the QAP is further transformed to a *quadratic knapsack problem*. Then an approach proposed by Gallo et al. [67] is used for computing lower and upper bounds for the latter problem. Unfortunately, in the case of quadratic knapsack problems arising from QAPs the gap between this bounds is quite large.
In the second approach the polyhedral approach proposed by Barahona and Mahjoub [8] is applied to the max cut problem with additional constraints obtained by the corresponding QAP transformation. Unfortunately, the resulting max cut problems are very large and the additional constraints seem to complicate the matter quite a lot.

2.4.2 Lower bounds

Lower bounding is the most studied topic on the QAP. Although a lot of efforts have been done to derive tight and computationally efficient lower bounds, such bounds are not found yet. Lower bounds are a component of crucial importance in branch and bound techniques. Moreover, they also are a basic tool for testing the quality of the solutions produced by heuristics. When considering branch and bound methods, both the tightness of the bounds and the time complexity of their computation are important. In the case that the lower bounds are used for heuristics evaluation the emphasis is on their tightness rather than on the time complexity of their computation. In this subsection we shortly describe the major bounding techniques for QAPs. They can be classified in three main groups: *combinatorially based techniques*, *reformulation techniques* and *algebraic techniques*.

Combinatorially based bounding techniques

Gilmore-Lawler bound (GL). One of the first lower bounds for QAPs proposed in the literature [72, 115] are the so called *Gilmore-Lawler bounds*. Consider a QAP(A,B) of size n with coefficient matrices $A = (a_{ij})$ and $B = (b_{ij})$. Consider a new $n \times n$ matrix $C = (c_{ij})$ defined as follows:

$$c_{ij} = \min_{\pi \in \mathcal{S}_n} \sum_{k=1}^{n} a_{i\pi(k)} b_{jk}, \qquad 1 \leq i, j \leq n$$

Each of the entries c_{ij} can easily be calculated by sorting vectors $A_{(i.)}$ and $B_{(j.)}$ increasingly and decreasingly, respectively (cf. Proposition 1.1). It takes $O(n^3)$ time to calculate all c_{ij}. Now, it is easily seen that the following equalities and inequalities hold, for each $\pi \in \mathcal{S}_n$:

$$Z(A, B, \pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij} = \sum_{i=1}^{n} \langle A^{\pi}_{(i.)}, B_{(i.)} \rangle \geq \sum_{i=1}^{n} c_{\pi(i)i}$$

The last inequality implies

$$\min_{\pi \in \mathcal{S}_n} Z(A, B, \pi) \geq \min_{\pi \in \mathcal{S}_n} \sum_{i=1}^{n} c_{\pi(i)i} = GL(A, B)$$

Thus, $GL(A, B)$ is a lower bound for the optimal value of QAP(A,B). $GL(A, B)$ is called the Gilmore-Lawler bound for QAP(A,B). Once the entries of matrix C have been calculated, it takes $O(n^3)$ time to calculate $GL(A, B)$ by solving a linear assignment problem.

These bounds are among the simplest and cheapest bounds to compute, but unfortunately they are not tight. Moreover, the gap between the Gilmore-Lawler bound and the optimal solution increases quite fast as the size of the problem increases.

The quality of GL bounds can be improved by transforming the coefficient matrices A and B to new matrices $\bar{A}$ and $\bar{B}$ with smaller entries. Then, the quadratic term $Z(\bar{A}, \bar{B}, \pi)$ would generally get smaller values then $Z(A, B, \pi)$, $\pi \in \mathcal{S}_n$. This process yields an additional term in the objective function. This term clearly depends on the permutation π. The crucial point is to use only such transformations of A, B into $\bar{A}$, $\bar{B}$, respectively, which lead to a *linear* additional term. In this case, the minimal value of the additional term over all permutations $\pi \in \mathcal{S}_n$ can be exactly computed. These kind of approaches are generally termed *reduction methods*. They were originally introduced by Conrad [46] and further studied by several authors [21, 22, 52, 59, 63, 157]. Generally a reduction approach transforms the coefficient matrices $A = (a_{ij})$ and $B = (b_{ij})$ in new matrices $\bar{A} = (\bar{a}_{ij})$ and $\bar{B} = (\bar{b}_{ij})$ by using formulas of the form

$$(\bar{a}_{ij}) = a_{ij} - f_j \quad 1 \le i, j \le n, \ i \ne j$$

$$(\bar{b}_{ij}) = b_{ij} - e_j \quad 1 \le i, j \le n, \ i \ne j$$

After this transformation one gets the so called *reduced* QAP with a linear term in its objective function as in (2.3). For the quadratic term in the objective function of the reduced QAP the GL bound can be then computed, whereas for the linear term the optimal (minimal) value can exactly be computed. The sum of these two values is an obvious lower bound for the reduced QAP and therefore, also for the original QAP. Clearly, different reduction schemes, i.e. different choices of vectors (f_i) and (e_i) above, yield GL bounds of different quality. There is no clear evidence showing that some reduction scheme is better than the others. Frieze and Yadegar [63] have shown that the best of these GL bounds equals the optimal value of a Lagrangean relaxation of (2.10) with best possible values of Lagrangean multipliers. They use two different subgradient approaches for finding approximate values for the best Lagrangean multipliers and get two new lower bounds, denoted by FY1 and FY2, which are generally better than GL bounds (even after reduction). Resende, Ramakrishnan and Drezner [154] tried to exploit the theoretical result achieved by Frieze and Yadegar for deriving better lower bounds. Namely, Frieze and Yadegar notice that for each QAP(A,B) the lower bound obtained as solution of an LP relaxation of (2.10) is at least as good as each GL bound obtained after some reduction. In [154] an interior point method is used for solving the LP relaxation proposed by Frieze and Yadegar. It turns out that the resulting lower bounds are among the best bounds for problems from QAPLIB [31], but they are very expensive (in terms of computation time). It often takes several hours to find lower bounds for instances of size 15!
Finally, it is worthy to notice that, given a QAP(A,B), checking whether the corresponding Gilmore-Lawler bound equals the optimal solution or not is an NP-complete problem (cf. [119]).

Christofides and Gerrard bound (CG). Another, more general, combinatorial bounding procedure is proposed by Christofides and Gerrard in 1981 [42]. This

bounding procedure is based on the following graph theoretical interpretation of the QAP. Consider a QAP(A,B) of size n. A and B are considered to be the weighted adjacency matrix of two complete graphs G_1 and G_2 on n vertices. Obviously, each $\pi \in \mathcal{S}_n$ is an isomorphism between G_1 and G_2. $Z(A, B, \pi)$ is considered to be the *cost* of this isomorphism. Solving QAP(A,B) means finding an isomorphism between G_1 and G_2 with minimum cost. The basic idea of Christofides and Gerrard consists of decomposing the graphs G_1 and G_2 into isomorphic subgraphs $G_1^{(1)}, G_1^{(2)}, \ldots, G_1^{(k)}$ and $G_2^{(1)}, G_2^{(2)}, \ldots, G_2^{(k)}$, respectively. The subgraphs $G_1^{(i)}$ $\left(G_2^{(i)}\right)$ should have the same vertex set as G_1 (G_2) and moreover, every edge e of G_1 (G_2) should occur in the same number of subgraphs $G_1^{(i)}$ $\left(G_2^{(i)}\right)$ and in at least one of them, for $1 \leq i \leq k$. An isomorphism between G_1 and G_2 maps each subgraph $G_1^{(i)}$ to exactly one subgraph $G_2^{(j)}$, $1 \leq i, j \leq k$. Let us denote the cost of such a mapping by c_{ij}. Clearly, the optimal value of the linear assignment problem with cost matrix $C = (c_{ij})$ is a lower bound for the optimal value of QAP(A,B). This bound is denoted by CG.
Obviously, this approach makes sense only in the case that the costs c_{ij} can be computed in polynomial time. For the sake of efficiency, the size of the linear assignment problem to be solved at the end of the bounding procedure should be $O(n^2)$. These two requirements limit the choice of the decomposition subgraphs. Examples of good decomposition subgraphs are stars, double stars and graphs consisting of a single edge [42]. It is worthy to notice that applying the CG bound for a decomposition into stars leads to the GL bound. Applying the CG bound for a decomposition into graphs consisting of single edges leads to a bound proposed by Land [111] and by Gavett and Plyter [69]. A decomposition into doublestars leads to tighter bounds than the other decompositions, but it is very expensive (time complexity $O(n^7)$).

Bounding techniques based on reformulations

First of all, we remark that lower bounds of this class are generally applied to QAPs formulated as in (2.2) and (2.4). Obviously, they can be also applied to a QAP(A,B). However, in this case, they cannot take advantage of the specific structure of the objective function coefficients.

A *reformulation* of a given QAP instance is another QAP instance, with other coefficients, such that each permutation leads to equal objective function values for both QAPs. The basic idea of these bounding techniques is to derive a sequence of reformulations $P_0 = P, P_1, \ldots, P_k$ for a given problem P by applying some appropriate reformulation rule. To each of the reformulations P_i, $0 \leq i \leq k$, another bounding technique is applied, for example the GL bound, and a lower bound, say B_i, is derived. The reformulation rule is "appropriate", in the sense that the sequence of lower bounds B_i is monotonically nondecreasing: $B_0 \leq B_1 \leq \ldots \leq B_k$. Usually, the construction of a new reformulated problem by applying the reformulation rule exploits the previous reformulated problems and the bounds obtained for them. Such bounding strategies have been proposed by Carraresi and Malucelli [36] and Assad and Xu [4]. The corresponding bounds are denoted by CM and AX, respectively. Both approaches use the GL bound for bounding in each iteration and produce bounds of good quality. However, these bounding techniques are quite time consuming, as n^2+1 linear assignment problems per iteration are solved. Moreover, when computing bounds CM and AX for a QAP(A,B), the nice structure of the objective function coefficients is not preserved by the reformulation rules. That is, even if in the above sequence of reformulations P_0 is a QAP(A,B), the problems P_i, $i \geq 1$, are QAPs of the form (2.2) or (2.4). Notice that the computation of the bound GL for a QAP as defined in (2.2) or (2.4), takes $O(n^5)$ time, whereas for QAP(A,B) it takes only $O(n^3)$. Thus, CM and AX are computed in $O(kn^5)$ time, where k is the number of iterations (reformulations).

It is reasonable to choose GL as bound to be applied in each reformulation, because it is cheap and easy to compute. However, in principle, any other bound can be involved in such a reformulation scheme.

Summarizing, the reformulation bounding techniques yield good bounds, but they are quite time consuming.

Algebraic bounding techniques

Eigenvalue related bounds. Among the algebraic bounding techniques the *eigenvalue* approach is the most celebrated. These approach cannot be applied to the general problems (2.2) and (2.4). In the case of QAPs as in (2.3) the eigenvalue related approaches can applied to the quadratic term of the objective function, whereas the optimal (minimal) value for the linear part can be exactly computed.

Eigenvalue based bounds for QAP(A,B), with A and B being symmetric matrices, are proposed by several authors in a series of papers [59, 83, 85, 152].

Bound EV. Consider the trace formulation of QAP in (2.6). The matrices A and B are assumed to by symmetric in order to guarantee their eigenvalues to be reals. Let $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$ be the eigenvalues of matrix A and $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_n$ be the eigenvalues of matrix B. Then, there exist orthogonal matrices P_1 and P_2 and diagonal matrices $\Lambda_1 = diag(\lambda_1, \lambda_2, \ldots, \lambda_n)$ and $\Lambda_2 = diag(\mu_1, \mu_2, \ldots, \mu_n)$, such that $A = P_1\Lambda_1 P_1^t$, $B = P_2\Lambda_2 P_2^t$. As shown in [59], the following equality holds:

$$tr(AXBX^t) = \lambda^t S(X)\mu \ , \quad \forall X \in \Pi \ , \tag{2.12}$$

where $S(X) = (\langle P_{1(.i)}, XP_{2(.j)}\rangle^2)$ and $P_{1(.i)}$, $P_{2(.i)}$ are the column vectors of matrices P_1 and P_2 and also the eigenvectors of A and B, respectively. The following inequality establishes the first eigenvalue related bound for QAP(A,B), proposed by Finke et al. in 1987, [59].

$$\sum_{i=1}^{n} \lambda_i\mu_i \leq tr(AXBX^t) \leq \sum_{i=1}^{n} \lambda_{n-i+1}\mu_i \ , \quad \text{for all } X \in \Pi \tag{2.13}$$

The bound $\sum_{i=1}^{n} \lambda_i \mu_i$, denoted by EV, usually is very poor because of the large variance of the eigenvalues λ_i, μ_i, $1 \le i \le n$. Most of the time the EV bound is found to be negative.

Bound EV1. One possibility to sharpen the EV bound is trying to decrease the difference $\sum_{i=1} \lambda_{n-i+1}\mu_i - \sum_{i=1} \lambda_i \mu_i$. This can be done by reducing the *spreads* of matrices A and B, where the spreads $sd(A)$, $sd(B)$ are defined as follows:

$$sd(A) = \max\{\lambda_i : 1 \le i \le n\} - \min\{\lambda_i : 1 \le i \le n\} = \lambda_n - \lambda_1$$

$$sd(B) = \max\{\mu_i : 1 \le i \le n\} - \min\{\mu_i : 1 \le i \le n\} = \mu_1 - \mu_n$$

There exist no simple formulas for determinating the spread of a given matrix, but there is a simple formula which gives an upper bound for it. Following these ideas, a successful attempt to improve the EV bound is the reduction (in contrast with its poor role in improving the GL bound). The coefficient matrices A and B are replaced by new symmetric matrices $\bar{A}$ and $\bar{B}$ with smaller spreads, respectively. In [59] the following reduction formulas are used,

$$\bar{A} = A - F_1 - F_1^t - D_1\,, \qquad \bar{B} = B - F_2 - F_2^t - D_2, \tag{2.14}$$

where F_1, F_2 are constant column matrices, F_1^t, F_2^t are their transposed matrices respectively, and D_1, D_2 are diagonal matrices. F_1, D_1, F_2 and D_2 are appropriately chosen in order to minimize the upper bounds on the spreads $sd(A)$ and $sd(B)$, respectively. F_1, D_1, F_2 and D_2 can be easily computed by using explicit formulas (see [59]). This reduction leads to a new bound EV1, which is definitely better than the EV bound especially for QAPs of size larger than 20.

Bound EV2. If the matrices A and B have zero diagonal elements, the EV1 bound can be further improved. Namely, after the reduction the objective function looks as follows:

$$Z(A, B, \pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} \bar{a}_{\pi(i)\pi(j)}\bar{b}_{ij} + \sum_{i=1}^{n} c_{\pi(i)} d_i \tag{2.15}$$

where $c_i = f_{ki}$ for any $k \in \{1, 2, \ldots, n\}$ and $d_i = \sum_{j=1}^{n} b_{ij}$. Denoting $C = (c_i)$ and $D = (d_i)$, we have $EV1 = \langle \bar{\lambda}, \bar{\mu}\rangle_- + \langle C, D\rangle_-$. Intuitively, separately bounding the quadratic and the linear part of the objective function as done by EV1, cannot always yield good bounds. One idea for a compromising lower bound is given by Rendl [149]. He introduces the so called *measure of linearity* L for the QAP(A,B) as follows:

$$L = \Big(\big\langle \bar{\lambda}, \bar{\mu}\big\rangle_+ - \big\langle \bar{\lambda}, \bar{\mu}\big\rangle_-\Big) / \Big(\langle C, D\rangle_+ - \langle C, D\rangle_-\Big)$$

A small L indicates a small influence of the quadratic term in (2.15) with respect to the linear term. In this case it is suggested to rank the scalar products $\langle C^\pi, D\rangle$, $\pi \in \mathcal{S}_n$, in increasing order and compute the objective function values corresponding to the respective permutations. Assume that the permutations π_i, $1 \le i \le k$, yield the k best values of the scalar product $\langle C^{\pi}, D\rangle$ over $\mathcal{S}_n$. Rendl [149] proves that the following inequalities hold:

$$Z(A,B,\pi_1) \geq Z(A,B,\pi_2) \geq \ldots \geq Z(A,B,\pi_k) \geq \langle \lambda, \mu\rangle_- + \langle C^{\pi_k}, D\rangle \geq \ldots \geq$$
$$\langle \lambda, \mu\rangle_- + \langle C^{\pi_1}, D\rangle$$

Moreover, if $Z(A,B,\pi_i) > \langle \lambda, \mu\rangle_- + \langle C^{\pi_i}, D\rangle$ then $\langle \lambda, \mu\rangle_- + \langle C^{\pi_i}, D\rangle$ is a lower bound to QAP(A,B) (obviously at least as good as EV1), otherwise π_i in an optimal solution to QAP(A,B). Finding the k-best permutations to minimize the scalar product $\langle C^{\pi}, D\rangle$ over $\mathcal{S}_n$ takes $O(n\log n + (n+\log k)k)$ time [149]. The bounds produced by this approach are much better than EV1 for matrices A and B whose row sums take a relatively large number of different values.
A large value of L indicates a negligible linear term in (2.15). In this case, an improvement for the eigenvalue bound $\langle \bar{\lambda}, \bar{\mu}\rangle_-$ of the quadratic term is obtained as follows. The quadratic term of the objective function in (2.15) can be written as

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \bar{a}_{\pi(i)\pi(j)}\bar{b}_{ij} = tr(\bar{A}X_{\pi}\bar{B}X_{\pi}^t) = \sum_{i=1}^{n}\sum_{j=1}^{n} \bar{\lambda}_i\bar{\mu}_j\langle U_i, X_{\pi}V_j\rangle^2 ,$$

where X_{π} is the permutation matrix corresponding to π, $\bar{\lambda}_i$, $\bar{\mu}_i$, $1 \leq i \leq n$, are the eigenvalues of matrices $\bar{A}$ and $\bar{B}$ (appropriately sorted) and U_i, V_i, $1 \leq i \leq n$, are their eigenvectors, respectively. The entries $\langle U_i, X_{\pi}V_j\rangle^2$ may bounded by $\ell_{ij} \leq \langle U_i, XV_j\rangle^2 \leq u_{ij}$, where

$$u_{ij} = \max\left\{\langle U_i, XV_j\rangle_-^2, \langle U_i, XV_j\rangle_+^2\right\} \quad \text{and}$$

$$\ell_{ij} = \begin{cases} 0 \text{ if } \langle U_i, XV_j\rangle_-,\ \langle U_i, XV_j\rangle_+ & \text{have different signs} \\ \min\left\{\langle U_i, XV_j\rangle_-^2, \langle U_i, XV_j\rangle_+^2\right\} & \text{otherwise} \end{cases}$$

Now, replacing $\left\langle \bar{\lambda}, \bar{\mu}\right\rangle$ in EV1 by the optimal value of the following capacitated transportation problem, yields the improved lower bound EV2:

$$\begin{array}{ll} \min & \sum_{i=1}^n \sum_{j=1}^n \bar{\lambda}_i\bar{\mu}_j x_{ij} \\ \text{subject to} & \\ & \sum_{i=1}^n x_{ij} = 1 \\ & \sum_{j=1}^n x_{ij} = 1 \\ & \ell_{ij} \leq x_{ij} \leq u_{ij} \end{array}$$

Other eigenvalue related bounds. The reduction used to improve bound EV in order to get bound EV1 is performed independently for the matrices A and B and the linear term of the reduced QAP is then bounded separately from the quadratic term. A way to obtain better bounds is to consider all these factors jointly. This idea is developed in Rendl and Wolkowicz [152].

Consider QAP(A,B) and the generic reduction formula in (2.14). Obviously, matrices $\bar{A}$, $\bar{B}$ and also matrix $\bar{C}$ defining the linear term of the reduced QAP depend on matrices F_1, F_2, D_1 and D_2. As matrices F_i, $i = 1, 2$, have constant columns and as matrices D_i, $i = 1, 2$, are diagonal matrices, the above generic reduction is fully defined by a $4n \times 1$ parameter vector. Let us denote it by ?. Let us denote the minimal scalar product of the eigenvalues of $\bar{A}$, $\bar{B}$, obtained by the reduction with parameter ?, by $msp(?) := \left\langle \bar{\lambda}, \bar{\mu} \right\rangle_-$. Moreover, denote

$$lap(?) = \min_{\pi \in \mathcal{S}_n} \sum_{i=1}^{n} \sum_{j=1}^{n} \bar{c}_{\pi(i)i} \,, \tag{2.16}$$

for $\bar{C}$ obtained by the reduction with parameter ?. Clearly, for each parameter vector ?, $msp(?) + lap(?)$ is a lower bound for the QAP(A,B). The natural goal is to find a parameter vector ? which maximizes this bound, i.e.,

$$\max \left\{ msp(?) + lap(?) \colon ? \in R^{4n \times 1} \right\} \tag{2.17}$$

$msp(?)$ is a differentiable function (at least when the eigenvalues have multiplicity 1), whereas $lap(?)$ is piecewise linear concave function. In [152], problem (2.17) is solved by using a steepest ascent algorithm.
This procedure often produces the best bounds for QAPs with symmetric coefficient matrices. However, it is quite time consuming as an EV bound must be computed in each iteration and a large number of iterations is needed to solve (2.17).

Another bounding idea related to the eigenvalues of the coefficient matrices arises when considering relaxations of the QAP formulated as in (2.6). It is well known that the class of permutation matrices is the intersection of three matrix classes: the class of *orthogonal matrices*, the class of matrices with *columns and rows sums equal to* 1 and the class of matrices with *nonnegative entries*. (The intersection of the last two classes yields the *doubly stochastic* matrices.) Minimizing the objective function of the problem (2.6) over one of these three classes of matrices or over intersections of pairs of them would yield relaxations of the QAP. The relaxation of (2.6) on the class of orthogonal matrices has been proven to be useful. Namely, Rendl and Wolkowicz [152] show that the EV bound is actually the optimal solution of the relaxation of the QAP (2.6) on the class of orthogonal matrices. The QAP relaxation on the class of orthogonal matrices which additionally have rows and columns sums equal to 1 is studied by Hadley et al. [85]. The constraint on the sums over the rows and columns of matrix X are equivalently reformulated to obtain a projected problem. The latter is an $n-1$ dimensional minimization problem which is equivalently reformulated a QAP relaxation on the set of orthogonal matrices. But this relaxation can be solved to optimality by an eigenvalue approach as noted above. Recently, the relaxation of QAP (2.6) on the class of orthogonal matrices with rows and columns sums equal to zero was further studied in relationship with trust region problems [98].

All eigenvalue related bounds described up to now work only for symmetric QAPs. If only one matrix is symmetric, the other one can be symmetricized yielding an equivalent QAP. Hadley, Rendl and Wolkowicz [84] propose also a technique to transform an asymmetric QAP to a QAP with (complex) Hermitian coefficient matrices. Then, an eigenvalue type bound analogous to EV is obtained for the transformed QAP (and equivalently for the given QAP) by applying the Hoffman-Wielandt inequality for Hermitian matrices.

Variance reduction based lower bounds. This class of lower bounds, proposed recently by Li, Pardalos, Ramakrishnan and Resende [119], is based on optimal reduction schemes for the QAP. Consider a QAP(A,B) of size n and a partition of A and B as $A = A_1 + A_2$, $B = B_1 + B_2$, where $A_1 = \left(a_{ij}^{(1)}\right)$, $A_2 = \left(a_{ij}^{(2)}\right)$, $B_1 = \left(b_{ij}^{(1)}\right)$ and $B_2 = \left(b_{ij}^{(2)}\right)$. For each pair (i,j), $1 \le i,j \le n$, denote by l_{ji} the solution of the following minimization problem:

$$l_{ji} = \min_{\substack{\pi\in\mathcal{S}_n\\ \pi(i)=j}} \left\{ \sum_{k=1}^n a_{j\pi(k)}^{(1)} b_{ik}^{(1)} + \sum_{k=1}^n a_{\pi(k)j}^{(2)} b_{ki} + \sum_{k=1}^n a_{\pi(k)j} b_{ki}^{(2)} - \sum_{k=1}^n a_{\pi(k)j}^{(2)} b_{ki}^{(2)} \right\} \tag{2.18}$$

The key observation is that the solution of the following linear assignment problem

$$\min_{\pi\in\mathcal{S}_n} \sum_{i=1}^n l_{\pi(i)i}$$

is a lower bound for QAP(A,B) (cf. [119]):

$$\min_{\pi\in\mathcal{S}_n} \sum_{i=1}^n l_{\pi(i)i} \le \sum_{i=1}^n \sum_{j=1}^n a_{\pi(i)\pi(j)} b_{ij} \tag{2.19}$$

Obviously, different choices for A_1, A_2, B_1 and B_2 yield different lower bounds. The GL bound is obtained from (2.19) for $A_1 = A$, $A_2 = (0)$ and $B_1 = B$ and $B_2 = (0)$, i.e., by applying the new bounding approach without partitioning the matrices A and B. The partitions considered in [119] are such that the *variances* of matrices A_1, A_2, B_1, B_2 and the averages of their *row variances* are minimized. For an $n \times m$ matrix M, its *average* $\gamma(M)$ and its *variance* $V(M)$ are defined as follows:

$$\gamma(M) := \sum_{i=1}^n \sum_{j=1}^m m_{ij} \qquad V(M) := \frac{1}{nm} \sum_{i=1}^n \sum_{j=1}^m \left(\gamma(M) - m_{ij}\right)^2$$

The *total variance* $T(M,\lambda)$ of matrix M depends on the parameter λ and is defined by

$$T(M,\lambda) := \lambda \sum_{i=1}^n V(M_{(i.)}) + (1-\lambda)V(M)$$

where $M_{(i.)}$ is the i^{th} row vector of M considered as a $1 \times m$ matrix. Thus, the total variance of M is a convex combination of the sum of the row variances of M with the variance of M. Considering partitions of the type $A_1 = A + \Delta$, $A_2 = -\Delta$ and trying to minimize the total variances of matrices A_1 and A_2, we get the following minimization problem:

$$\min_{\Delta \in \mathbb{R}^{n\times n}} \theta T(A+\Delta,\lambda) + (1-\theta)T(-\Delta^t,\lambda) \tag{2.20}$$

for fixed parameters $0 \le \theta \le 1$ and $0 \le \lambda \le 1$. Two approximate solutions are proposed for (2.20) giving the entries δ_{ij} of matrix Δ.

($\mathbf{R_1}$) $\delta_{ij} = \theta(a_{nn} - a_{ij}) + \delta_{nn}, \qquad i,j = 1,\ldots,n.$

($\mathbf{R_2}$) $\delta_{ij} = \theta\left(\gamma(A_{(.n)}) - \gamma(A_{(.j)}))\right) + \delta_{nn}, \qquad i,j = 1,\ldots,n.$

Obviously, these solutions depend on the control parameter θ. When applying reduction scheme (R_1) with $\theta = 0$ one gets the bound GL. Experimental results have shown that the best choices for the parameter θ are: $\theta = 0$, when applying (R_1), and $\theta = 1$, when applying (R_2). Matrix B is partitioned in an analogous way.
In order to efficiently compute these bounds, the entries l_{ij} are replaced by lower bounds to the optimal value of problem (2.18). The computation of the new lower bounds takes then $O(n^3)$ elementary steps. It has been experimentally proven that these bounds, which are fast to compute, are by far better than the Gilmore-Lawler bounds and competitive with the other "cheap" bounds[1]. For problems with high variance of coefficient matrices, the new bounding procedure yields almost always the best existing lower bounds.

Bounds based on semidefinite relaxations. These bounds are based on quadratic (0-1) programming formulations for the QAP. Semidefinite relaxations to these formulations are considered and are handled by standard methods, e.g. cutting planes method. The solutions of these relaxations clearly provide lower bounds for the given QAP. For a detailed description of semidefinite relaxations for the QAP we refer to [97]. In [97] it is shown that these approaches produce very good bounds, especially when the size of the problem increases. However these bounds are extremely time consuming to compute.

2.5 Heuristics

We saw in the previous section that only small QAP instances (instances of size at most 20) can be solved to optimality. On the other side, the large spectrum of QAP applications often leads to instances of much larger size. Under these conditions, polynomial time heuristics providing suboptimal solutions to the QAP abound in the literature. Without pretending to mention all kinds of heuristic approaches which

[1]Bounds which can be computed within $O(n^3)$ time are considered as "cheap".

have been applied to QAPs, we try to review the main and most fruitful ideas in this research direction. There are five main streams of heuristic approaches to QAP, listed in a chronological order here below.

1. Construction methods
2. Limited enumeration methods
3. Improvement methods
4. Simulation approaches
5. Genetic algorithms
6. Greedy randomized search

Construction methods

Construction methods are considered to be the easiest heuristic approaches to the QAP, from both the conceptual and the implementation point of view. This simplicity is often associated with a quite poor quality of the corresponding solutions. Basically, all construction methods start with an empty partial solution (permutation) and recursively assign locations to facilities according to certain criteria, until all facilities have been assigned. These methods are probably the oldest ones, dating back to the early 60s with a heuristic proposed by Gilmore [72]. A refined version of the typical construction method introduced by Gilmore was proposed by Burkard [22]. Another construction method which yields relatively good results is proposed by Müller-Merbach [135]. This is the so called *increasing degree of freedom* method. It starts with the empty partial permutation and with a fixed order of indices $1, 2, \ldots, n$, say $r_1, r_2, \ldots, r_n$, where n is the size of the considered QAP. Denote by M_π the set indices in $\{1, 2, \ldots, n\}$ which are already assigned by the current partial permutation π. Moreover, denote $\pi(M) = \{\pi(i) : i \in M\}$, for some $M \subset \{1, 2, \ldots, n\}$. For a current partial permutation π with $M_\pi = \{r_1, r_2 \ldots, r_{k-1}\}$ a new permutation π', with $M_{\pi'} = M_\pi \cup \{r_k\}$, is constructed as follows. First, assign r_k to a $j \notin \pi(M_\pi)$ and compute the corresponding increase on the objective function δZ_j. Then consider assigning r_k to an index $j \in \pi(M_\pi)$. Let $r_i \in \{r_1, r_2, \ldots, r_{k-1}\}$ such that $\pi(r_i) = j$. Denote by δZ_{jl} the change in the objective function when assigning r_k to j and r_i to l, for some $l \notin \pi(M_\pi)$. Among these $k(n-k+1)$ possibilities for constructing π', choose the one which yields the smallest increase or change in the objective function, respectively. Then repeat this procedure until all indices r_i, $1 \le i \le n$ are assigned to some indices in $\{1, 2, \ldots, n\}$.

Recently, Murthy and Pardalos [133] proposed a construction method based on the bound GL. Numerical experiments on some well known test problems show that the relative error of suboptimal solutions provided by this heuristics is about 15%.

Limited enumeration methods

Limited enumeration methods are actually strongly related to exact methods such as branch and bounds and cutting planes. Limited enumeration methods based on branch and bound algorithms basically rely on the fact that an optimal or a very good solution is very often found in an early stage of the search, whereas the rest of the time is spent for proving the optimality of this early found solution or improving it, respectively. How to take advantage from this behavior, in order to produce good solutions to the QAP (solutions which are either optimal or close to the optimum) in a reasonable time?

A first possibility are the so called *time limits*. We can stop the enumeration process either after a prespecified time limit has been reached or in the case that no improvement has been made during a time interval longer than a prespecified one. Obviously the prespecified parameters depend on the problem size.

A second possibility would be to decrease the upper bound. For example, if no improvement has been found after a certain prespecified time interval, then the upper bound is decreased by a certain percentage. This approach may obviously cut off the optimal solution, but in any case it speeds up the search. Moreover, the suboptimal solution differs from the optimal one not more than the above mentioned percentage.

Methods based on statistical considerations to evaluate the value of the objective function can be also classified in this group of heuristics. For instance, Graves and Whinston [79] calculate the expectation and the variance of the objective function value of a QAP solution obtained when completing a given partial permutation. Combining these ideas with a pair-exchange improvement method, West derives a quite powerful heuristic [183].

Heuristics arising when applying cutting plane methods are other interesting approaches which can be classified in this group. Heuristics of this type have been developed, for example, by Bazaraa and Sherali [13] and Burkard and Bönniger [23]. In [13] the QAP is equivalently formulated as a mixed integer program and then a Benders' decomposition [15] is applied to it. One of the resulting problems (the *slave* problem) is a transportation problem and can be optimally solved. The other problem (the *master* problem) is a $0-1$ linear program, which is solved by a cutting plane method. A heuristic approach is derived by applying a limited number of cuts along with Benders' scheme. Although this approach is quite sensitive towards the initial solution, it yields suboptimal solutions of good quality. Its most significant drawback are large memory requirements, which sometimes cause the failure of the method (when used as exact algorithm), rather then running time requirements. Burkard and Bönniger in [23] use no Benders' decomposition. They apply a cutting plane type method to a $0-1$ linear program formulation of QAP proposed by Balas and Mazzola [7]. The running time of the heuristic is controlled by the number of iterations it runs through, where two linear assignment problems are solved in each iteration. In contrast with the approach described in [13], this method seems to be quite stable, in the sense that the quality of the resulting suboptimal solutions only slightly depends on the initial ones. From the point of view of solution quality both methods are comparable, whereas the latter is advantageous with respect to memory and running time requirements.

Improvement methods

Improvement methods are classical approaches used to difficult combinatorial optimization problems. Most of heuristic methods for QAPs fall into this group. *Local search* algorithms and *tabu search* approaches can be distinguished as two major subgroups.

Local search methods. As described in the first chapter, a local search procedure starts with an initial feasible solution. It iteratively tries to find a better feasible solution in the neighborhood of the current one. This iterative step is repeated until no further improvement can be found. Thus, the definition of the neighborhood is a crucial point for this type of heuristics. Frequently used neighborhoods for QAPs are the so called *pair-exchanges* and *cyclic triple-exchanges* neighborhoods. In the case of pair-exchanges, the neighborhood of a given solution (permutation) consists of all permutations which can be obtained from the given one by applying a transposition to it. In this case, scanning the whole neighborhood would take $O(n^3)$ time, as the size of the neighborhood itself is $\binom{n}{2}$ and the objective function value $Z(A, B, \pi')$ for the neighbor π' of π can be calculated in $O(n)$ time, once the value $Z(A, B, \pi)$ is known. Moreover, if the neighborhood of π is already scanned and π' is a neighbor of π, then the neighborhood of π' can be scanned in $O(n^2)$. This is an easy but important result of Frieze et al. [64] which is always used when a complete neighborhood evaluation is required.

In the case of cyclic triple-exchanges, the neighborhood of a solution (permutation) π consists of all permutations obtained from π by a cyclic exchange of three indices. For example, if the triple of indices to be exchanged is (i, j, k), then the obtained permutation π' is defined as follows:

$$\pi'(x) = \begin{cases} \pi(x) & x \notin \{i, j, k\} \\ \pi(k) & x = i \\ \pi(i) & x = j \\ \pi(j) & x = k \end{cases}$$

Obviously, in this case the size of the neighborhood amounts to $2\binom{n}{3}$. This increase of the neighborhood size causes the corresponding local search approaches to be much more time consuming than the former ones. Moreover, the cyclic triple-exchanges do not really lead to considerably better results when compared with pair-exchanges.
Some efforts have been done on combining pair-exchanges and cyclic triple-exchanges. As an example, consider the results obtained by Mirchandani and Obata [130], where,

along with all pairwise exchanges, some three way and four way exchanges are evaluated. The important point here is that the size of the neighborhood remains $O(n^2)$.

Another question which arises along with local search algorithms is the order of the neighborhood scanning. This can be done either in a previously fixed order or in a randomly chosen order. Once the neighborhood structure and the order in which the neighborhood is scanned are fixed, several local search procedures can be distinguished depending on the criteria for updating the current feasible solution. In the case of pair exchange algorithms three are the most frequently used methods:

- Method of first improvement
- Method of best improvement
- Heider's method [90]

The method of *first improvement* updates the current solution as soon as the first improving neighbor solution is found. The method of *best improvement* scans the whole neighborhood and chooses the best improving neighbor solution if there is any, i.e., an improving neighbor solution with the smallest corresponding value of the objective function. *Heider's method* starts by scanning the neighborhood of the initial solution in a prespecified order. Actually, the order of the transpositions applied to the current solution for generating its neighbors is prespecified. The current solution is updated as soon as an improving neighbor solution was found. The neighbors of the new current solution are again generated by applying to it all transpositions in the prespecified order. This is done by starting with the successor of the last transposition applied to the previous current solution. The transpositions are ordered cyclically, that is the first transposition is the successor of the last one.
It is worthy to notice here a best improvement local search procedure with the $\mathcal{N}_{K-L}$ neighborhood structure, derived by Murthy, Pardalos and Li [134]. The method was tested on the Nugent test problems from QAPLIB and on some QAP instances with known optimal solution generated by using an algorithm described in [138]. (We will discuss this algorithm in the next section.) Although finding a local optimum to the QAP with this neighborhood structure is a PLS-complete problem, this heuristic found optimal solutions for most of the test problems. For the rest of tested instances it found solutions very near to the optimum, whereas running for less than two minutes in all tests, including instances of size 100.
In order to get better results, local search algorithms are performed several times with different initial solutions. They may be easily combined with construction algorithms, by running the construction algorithm to generate the initial solution. However, there are no clear suggestions on the choice or the construction of the initial solution. Bruijs [20] proved that generally, there is no strong argument in favor of good-quality initial solutions.

Tabu search approaches. Tabu search is proven to be a useful heuristic for solving hard combinatorial optimization problems in general and the QAP in particular. A detailed description of tabu search specific aspects and its applications to various optimization problems are given in [75].

It was introduced by Glover [73, 74] as a technique to overcome local optimality in combinatorial search. It involves as basical tools a *neighborhood structure*, a *move*, a *tabu list* and an *aspiration criterion*. A *move* is an operation which, when applied to a certain solution π, generates a neighbor π' of it. In the case of QAPs the moves are usually transpositions and the neighborhood is the pair-exchanges neighborhood. A *tabu list* is a list of forbidden moves, i.e., moves which cannot be applied to the current solution. Clearly, the tabu list is updated during the search procedure. Forbidden moves are also called *tabu moves*. An *aspiration criterion* is a condition which, when fulfilled by a tabu move, cancels its tabu status.
A generic tabu search procedure works as follows. It starts with an initial feasible solution and selects a best quality solution among (a part) of its neighbors obtained by non-forbidden moves. Note that this neighboring solution does not indispensably need to improve the objective function value. Then the current solution is updated and this procedure is repeated. Obviously, this procedure can *cycle*, i.e., visit some solution more than once. Trying to avoid this phenomenon, moves which, according to some criteria, are judged to lead to cycles, are added to the tabu list. As, however, forbidding certain moves could prohibit visiting interesting solutions, an aspiration criterion is introduced. It serves to distinguish the probably interesting moves among the forbidden ones. The search procedure stops when a stop criterion is fulfilled. The stop criterion often is a running time limit, or an iterations number limit.

One of the papers in which tabu search was applied to the QAP is written by Skorin-Kapov [169]. She used a tabu list of a fixed size, where this size is a control parameter. The appropriate value of this parameter very much depends on the problem instance. The mathematical nature of this dependence is unknown and consequently, it is quite difficult to tune this parameter. In [169], a move is declared as tabu if the transposition defining it has been applied for updating the current solution during the last ℓ iterations, where ℓ is another control parameter. The procedure stops after running a fixed number of iterations, where this number is the third control parameter. An important drawback of this algorithm consists on the difficulty of tunning its control parameters. The dependence of the latter on the problem instance is strong, but not clear. This hurts the robustness of the method.

Trying to overcome this difficulty, Taillard [174] proposes a new version of tabu search for the QAP, the so called *robust tabu search*. His version differs from the version of Skorin-Kapov in two main points. First, a move is declared tabu if it locates both interchanged facilities to locations they had occupied within the s most recent iteration, where s is the size of the tabu list. Second, the size of the tabu list is frequently changed by randomly choosing it between a minimal and a maximal value. These two values are control parameters. Numerical results show that this method produces quite good solutions when applied with $O(n^2)$ iterations to problems of size n and tabu list size in a range of 10% about the problem size. Moreover, this algorithm is more robust than the previous one, in the sense that its performance is less sensitive towards the control parameter values.

Recently, another version of tabu search, the so called *reactive tabu search*, was proposed by Battiti and Tecchiolli [10] and has produced quite good results when applied to QAPs. The reactive tabu search aims to a weaker dependence on the values of the control parameters. It involves a simple mechanism for adapting the tabu list size according to the properties of the considered problem instance. In principle, it notices when a cycle occurs, i.e., when a certain solution is revisited, and increases the tabu list size according to the length of the detected cycle. Once in a while the tabu list size is slightly reduced in order to keep it within reasonable limits. If, moreover, the number of solutions which are revisited a "large" number of times (this large number of times is a control parameter) exceeds a certain parameter called CHAOS (another control parameter) a random diversification of the search towards a new feasible solution is forced. Numerical results on test problems known in the literature show that, in most of the cases, reactive tabu search converges to the best known solution faster than any other tabu search scheme. For a comparison of different tabu search schemes and a genetic hybrid scheme (see [60]) for the QAP, the reader is referred to [175]. The author proposes a classification of the mostly studied QAP instances and suggests one out of the above mentioned approaches to be probably the best for the considered class of problems. Recently, also parallel implementations of tabu search have been proposed [39]. It is worthy to notice that this heuristic naturally allows parallelization by dividing the bargain of the neighborhood search among several processors.

Simulation approaches

Simulation approaches are another group of heuristics used for hard combinatorial optimization problem trying to overcome local optimality. They technically inherit the structure and some properties of local search algorithms, whereas in principle, they are based on the interesting analogy between problems in combinatorial optimization and in statistical mechanics. Kirkpatrick, Gelatt and Vecchi [104] developed the similarities between these two fields. They showed how the Metropolis algorithm [129] for simulating the behavior of a physical many-particle system, provides a natural tool for bringing techniques of statistical mechanics to bear in optimization. Trying to apply these ideas on the traveling salesman problem (TSP) they introduced the so called *simulated annealing*. The same method for the TSP was developed independently by Černỳ [38]. Burkard and Rendl [34] showed that simulated annealing is a general approach which can be applied to each combinatorial optimization problem which possesses a neighborhood structure. The following two points are basic for the analogy between a combinatorial optimization problem and a many-particle physical system:

- Feasible solutions of the combinatorial optimization problem correspond to states of the physical system.
- The objective function of a feasible solution corresponds to the energy of a state of the physical system.

In *condensed matter physics*, *annealing* is known as a thermal process for obtaining lower energy states of a solid in a heat bath.

The process runs through two phases. First, increase the temperature of the heat bath to a maximum value at which the solid melts. Secondly, decrease *carefully (slowly)* the temperature of the heat bath until the particles arrange themselves in the *ground state* of the solid. Notice that the ground state is characterized by a minimum of energy. Metropolis algorithm, which simulates the evolution of a solid in a heat bath in *thermal equilibrium*, is based on Monte Carlo techniques for generating a sequence of states of the solid. Let i be the current state with energy E_i. A subsequent state j, with energy E_j, is generated by applying a small perturbation to the current state, say displacing one of the particles. If the energy difference $E_j - E_i$ is negative, then state j is accepted as the next current state. Otherwise, j is accepted with a certain probability given by $\exp(\frac{E_i - E_j}{k_B t})$, where t denotes the temperature and k_B is the so called *Boltzmann constant*. If the temperature is decreased sufficiently slowly, the solid can reach thermal equilibrium at each temperature. In the Metropolis algorithm this is achieved by generating a large number of states at a given temperature value. The thermal equilibrium is characterized by the so called *Boltzmann distribution*, which gives the probability of the solid being in a state i with energy E_i, at temperature t:

$$P\{x = i\} = \frac{1}{Q(t)} \exp\left(\frac{-E_i}{k_B t}\right) ,$$

where x is a random variable denoting the current state of the solid. $P(t)$ is the so called *partition function* defined by

$$Q(t) = \sum_i exp(\frac{-E_i}{k_B t}) ,$$

where the summation extends over all possible states. The Boltzmann distribution plays an essential role in the theoretical analysis of the convergence of simulated annealing algorithms.

Simulated annealing, like tabu search, belongs to the group of the so called meta-heuristics, as it can be applied to any combinatorial optimization problem which possesses a neighborhood structure. In analogy to Metropolis algorithm, Figure 2.1 presents a generic simulated annealing algorithm for a generic combinatorial optimization problem defined by Definition 1.3. The routines used by this algorithm are described below. For two feasible solutions $X, Y \in \mathcal{F}$, $\Delta(X,Y)$ denotes the change in the value of the objective function when moving from X to Y, i.e. $\Delta(X,Y) = \sum_{y\in Y} f(y) - \sum_{x\in X} f(x)$. $move(X)$ is a function which returns a neighboring solution of X. Function g gives the value t_i of a parameter analogous to the temperature in the current iteration i, taking as argument its value t_{i-1} in the previous iteration. For each i, inequality $t_i < t_{i-1}$ holds. Moreover, as an essential requirement related to the choice of g, the equality $\lim_{i\to\infty} t_i = 0$ must hold. Finally, $random(0,1)$ generates a random number between 0 and 1 which serves to decide whether the current feasible solution will be updated by its recently generated neighbor or not. The generic simulated annealing algorithm presented in Figure 2.1 can

```
procedure Simulated annealing;
begin
  $X := X_0$; ($X_0$ is some initial starting point in $\mathcal{F}$)
  $i := 0$; (Initialize the iterations counter)
  $t_i := t_0$; ($t_0$ is a control parameter analogous to the initial temperature)
  repeat
    repeat
      $Y = move(X)$;

      If $\Delta(X,Y) < 0$, $accept := yes$;
      If $\Delta(X,Y) \geq 0$ and $\exp\left(\frac{-\Delta(X,Y)}{t_i Q(t_i)}\right) > random(0,1)$, $accept := yes$;
      If $accept = yes$ then $X := Y$;
    until equilibrium is approached sufficiently closely
    $i := i + 1$; (Update the iteration counter)
    $t_i = g(t_{i-1})$; (Update the current temperature)
  until a stop criterion is fulfilled ("the system is frozen")
end
```

Figure 2.1: The generic simulated annealing scheme

mathematically be modeled by an inhomogeneous ergodic Markov chain. Its transition probabilities, which are probabilities of moving from a feasible solution to some neighboring one, depend on the change on the objective function value implied by the corresponding move and on the current "temperature" value. Clearly, they indirectly depend on the neighborhood structure as well. The theory of Markov chains has been used for the analysis of the (probabilistic) convergence of simulated annealing algorithms. Some authors have derived conditions on the neighborhood structure, which guarantee the convergence of the generic algorithm to a (globally) optimal solution with probability equal to one, when the number of iterations approaches infinity, see for example [57]. Other authors essentially require a slow enough *cooling*, or more concretely $t_i = \frac{\Gamma}{\log i}$, where ? $\geq d$ and d is a constant depending on the combinatorial structure of the considered problem. For a detailed discussion on different theoretical aspects of simulated annealing methods, the reader is referred to [5, 110]. It remains an (apparently difficult) open problem to theoretically investigate the speed of the above mentioned asymptotic probabilistic convergence in general, and its practical impact to the performance of simulated annealing approaches for certain combinatorial optimization problems in particular.

Simulated annealing is proven to be a useful approach to optimization problems arising in computer design, wiring, component placements, as well as to traditional combinatorial optimization problems as partitioning, traveling salesman and QAP. Its main drawback is the relatively high number of control parameters and the absence of a widely accepted and well argued strategy for choosing their values. The first who derived a simulated annealing scheme for the QAP were Burkard and Rendl [34]. In [34] (and in all existing simulated annealing versions for the QAP) the pair-exchanges neighborhood structure is considered. The computational experiments in [34] confirm an expected behavior of the algorithm: its performance strongly depends on the temperature schedule. However, a careful tunning of the control parameters often yields high quality solutions. A more sophisticated simulated annealing approach for QAPs was proposed by Wilhelm and Ward [184]. The algorithm proposed in [184] uses a more appropriate definition of the equilibrium state, trying to find a better mathematical expression of the state of "thermal equilibrium". The authors report solutions of quite good quality, but they do not motivate their control parameters choices. Connolly [45] tries to analyze the role of the optional components of simulated annealing approaches as annealing scheme, random or sequential neighborhood search and temperature control. He introduces the so called *optimal temperature value*, that is a temperature state where most of the search should perform. This idea will be discussed more or less in details in Chapter 6, where three simulated annealing schemes for BiQAP are proposed. Laursen [114] argues on the choice of several control parameters and experimentally investigates the benefit of different strategies for setting their values.

What about comparing the performance of simulated annealing approaches with that of tabu search approaches to QAP? There is no general agreement in this point. Recently Battiti and Tecchiolli [11] compare a standard simulated annealing scheme with a reactive tabu search algorithm on a quite restricted set of QAP instances from QAPLIB. They conclude that in the long run both methods are competitive with respect to the quality of the produced solutions. When the running times increase, the reactive tabu search generally provides better solutions, whereas simulated annealing gets more easily stucked in local optima.

Genetic Algorithms

Another nature inspired approach to large scale combinatorial optimization problems are the so called *genetic algorithms*, shortly GA. The underlying motivation of such algorithms is the attempt to borrow ideas from the natural selecting process, which develops complex and well adapted species through relatively simple evolutionary mechanisms. The main point is to adapt these simple evolutionary mechanisms to combinatorial optimization problems. The first genetic algorithm was developed by Holland in 1975. Recently a lot of research has been done in deriving good GAs for combinatorial optimization problems. One reason of enhancing the research in this direction is the simplicity of parallel implementations for these kind of algorithms.

Basically, a genetic algorithm starts with a set of initial feasible solutions (generated randomly or by using some heuristic) called the *initial population.* The elements of a population are usually termed "individuals" or "members". The algorithm *selects* a number of pairs of *parents* from the current population and produces a new feasible solution out of each pair of parents, by using the so called *crossover rules*. Then, it throws a number of "bad" solutions out of the current population. This process is repeated until a *stop criterion* is fulfilled. The stop criterion may often be a time limit, an iterations number limit or the convergence of the algorithm, i.e., the current population consists of several copies of the same individual. During the run of the algorithm a *mutation* or an *immigration* is periodically applied to the current population trying to improve its overall quality by changing up some of the individuals or replacing them by better ones, respectively. Very often *local optimization* tools are also periodically used in order to improve the performance of the algorithm in general, and speed up its convergence in particular. For the diversification of the search the so called *tournaments* are used. In principal a tournament consists of applying several runs of GA starting from different initial populations and stopping them before they converge, respectively. Then derive a better population by combining the end-populations resulting from different runs and start a new GA run on it. A generic scheme of a genetic algorithm without tournaments is presented in Figure 2.2. For a very good coverage of theoretical and practical issues on genetic algorithms the reader is referred to [48, 76]. Several genetic algorithm approaches have been applied to the QAP. The algorithm developed by Tate and Smith [177], a more or less standard one, reveals some of the drawbacks of such algorithms, despite of encouraging numerical results. This algorithm does not find the best known solutions for Nugent problems of size 20 and 30. For larger problems of size up to 100, it cannot really compete with tabu search procedures. Trying to overcome these difficulties Fleurent and Ferland [60] propose so called hybrid approaches, derived by combining genetic algorithms with other heuristic procedures for QAP, particularly local search and tabu search. They manage to improve the well known solutions to most of the large scale test problems of Taillard and Skorin-Kapov (see QAPLIB). However, the time-cost of these improvements is very high, sometimes reaching more than 20 hours

```
procedure Genetic algorithms;
begin
  generate the initial population $\mathcal{P}$;
  i:=0; (Initializing the iteration counter)
    repeat
      repeat
        $i := i + 1$;
        select two individuals $X_1, X_2 \in \mathcal{P}$;
        apply the crossover operator on $X_1$ and $X_2$ to obtain child $X_3$;
        $\mathcal{P} := \mathcal{P} \cup \{X_3\}$;
      until $i$ has reached the prespecified value
      Applying some rule, generate $\mathcal{D} \subset \mathcal{P}$ with prespecified cardinality;
      Update $\mathcal{P} := \mathcal{P} \setminus \mathcal{D}$;
      Occasionally perform immigration and/or mutation;
      Occasionally perform local search;
    until a stop criterion is fulfilled
  end
```

Figure 2.2: The generic scheme of genetic algorithms

CPU time. Recently, Ahuja, Orlin and Tivari [2] obtained very promising results on large scale QAPs from QAPLIB, by applying the so called *greedy genetic algorithm*. The greedy genetic algorithm is tested on all problems from QAPLIB, obtaining the best known solutions for 103 out of 132 test problems. The solutions obtained for 28 out of the 29 remaining problems deviate less than 1% from the best known ones. These results are obtained by applying the algorithm only once and within reasonable computation time limits. The greedy elements incorporated in this algorithm seem to help maintaining the balance between biased search and diversity of the population.

Greedy randomized search

The single member of this group of heuristics is the so called GRASP (Greedy Randomized Adaptive Search Procedure) for QAPs [120]. GRASP is a combination of greedy elements with random search elements in a 2 phase heuristic consisting of a construction phase and a local improvement phase. In the construction phase GRASP starts by assigning a pair of facilities i_0, j_0 to a pair of locations j_0, k_0, respectively. According to the greedy component, GRASP selects this pair of assignment among those with minimal cost. That is, when considering a QAP(A,B) of size n, the greedy component tends to select i_0, j_0, k_0, l_0 such that

$$a_{i_0j_0}b_{k_0l_0} = \min\Big\{a_{ij}b_{kl}\colon i,j,k,l \in \{1,2,\ldots,n\},\quad i \neq j,\quad k \neq l\Big\}$$

The random element gives some freedom to the search procedure, with the hope to avoid getting trapped in poor locally optimal solutions. This works as follows. The non-diagonal entries of matrix $A = (a_{ij})$ are sorted increasingly, keeping only $\lfloor \beta(n^2 - n) \rfloor$ of them, for some $0 < \beta < 1$, where β is a control parameter. The non-diagonal entries of matrix $B = (b_{ij})$ are sorted decreasingly, keeping only $\lfloor \beta(n^2 - n) \rfloor$ of them. Then we get:

$$a_{i_1 j_1} \le a_{i_2 j_2} \le \ldots \le a_{i_{\lfloor \beta(n^2-n) \rfloor} j_{\lfloor \beta(n^2-n) \rfloor}}$$
$$b_{k_1 l_1} \ge b_{k_2 l_2} \ge \ldots \ge b_{k_{\lfloor \beta(n^2-n) \rfloor} l_{\lfloor \beta(n^2-n) \rfloor}}$$

The costs of possible pair assignments are then sorted increasingly, keeping the smallest $\lfloor \alpha\beta(n^2 - n) \rfloor$ among

$$a_{i_1 j_1} b_{k_1 l_1}, a_{i_2 j_2} b_{k_2 l_2}, \ldots, a_{i_{\lfloor \beta(n^2-n) \rfloor} j_{\lfloor \beta(n^2-n) \rfloor}} b_{k_{\lfloor \beta(n^2-n) \rfloor} l_{\lfloor \beta(n^2-n) \rfloor}} ,$$

where $0 < \alpha < 1$ is the second control parameter. In the next steps of the construction phase, facilities are assigned to locations, one facility to one location at a time. Assume we are performing the r$^{\text{th}}$ iteration of the construction phase and ? is the set of the already made assignments:

$$? = \{(i_1, k_1), (i_2, k_2), \ldots, (i_r, k_r)\}$$

Compute the cost of locating some facility j, $j \notin \{i_1, i_2, \ldots, i_r\}$, to some location l, $l \notin \{k_1, k_2, \ldots, k_r\}$, with respect to the already made assignments:

$$c_{jl} = \sum_{(ik) \in \Gamma} (a_{ij} b_{kl} + a_{ji} b_{lk})$$

If there are m unassigned feasible facility-location pairs, select at random one out of the $\lfloor \alpha m \rfloor$ best among them. Add the selected pair to ? and repeat this step until a permutation in $\mathcal{S}_n$ has been constructed. Here, the construction phase of GRASP terminates. The local improvement phase consists of a standard local search algorithm starting with the solution constructed in the first phase. The whole procedure is repeated a certain number of times, where this number is the third and last control parameter. In [120] GRASP is tested on most of the test problems from QAPLIB and on two classes of QAP instances with known optimal solution from [117]. (The generation of QAP instances with known optimal solution is the object of the next section.) The version of GRASP tested in [117] involves a local search approach with the pair-exchanges neighborhood, with $\beta = 0.1$, $\alpha = 0.5$ and maximum number of iterations equal to 100000. This algorithm yields best known solutions for most of the tested instances and even improves the best known solutions in a few cases. Moreover, its running times are reasonable, compared to other heuristic approaches.

Some results of recent experiments presented in [2] show that the greedy genetic algorithm mentioned in the previous subsection, outperforms GRASP, for the same running time. However, the same experiments show that in early stages of search GRASP produces better solutions, i.e., GRASP would be advantageous in the short run.

2.6 QAP instances with known optimal solutions

In this section we discuss two *generators of QAP instances with known optimal solutions*. The existence of such generators is very important, since instances with known optimal solutions can be used as test problems for the evaluation and comparison of heuristic approaches for QAPs.

Palubeckis' generator for QAPs with a known optimal solution

Here we describe one of the first methods for generating QAP instances with a known optimal solution, proposed by Palubeckis [138] in 1988. The input of the Palubeckis' algorithm consists of the size, say n, of the instance to be generated, the optimal solution (permutation) π for the output instance, the distance matrix A of an $r \times s$ rectangular grid, with $rs = n$, and two control parameters w, z, $z < w$. Here, rectilinear distances, termed also Manhattan distances, are considered. The algorithm starts with a trivial QAP instance with known optimal solution. Namely, it starts with $QAP(A, B^{(0)})$, where $B^{(0)} = (b_{ij}^{(0)})$ is a constant matrix, $b_{ij}^{0} = w$, $1 \le i, j \le n$. The identical permutation id is an optimal solution of the trivial problem $QAP(A, B^{(0)})$. The algorithm step by step transforms matrix $B^{(0)}$ to a matrix B which is not constant any more, whereas matrix A remains unchanged. This transformation is done so that permutation id remains an optimal solution to all intermediate QAP instances. In the last step the algorithm uses a simple standard trick for transforming the QAP instance with optimal solution id to another QAP instance with optimal solution π. A correct mathematical representation of Palubeckis' algorithm is given below.

Algorithm 2.1 *Palubeckis' generator for QAP instances with a known optimal solution.*

Input: Five positive natural numbers n, r, s, z, w such that $rs = n$, $z < w$ and a permutation $\pi \in \mathcal{S}_n$.

Output: Two $n \times n$ matrices A, B such that π is an optimal solution to QAP(A,B) and $w \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}$ is its optimal value.

```
Initialize  Input $n$, $r$, $s$, $z$, $w$ and $\pi$.
    Set $a_{ij} = |\lfloor j/r \rfloor - \lfloor i/r \rfloor| + |i (\mathrm{mod}\ r) - j (\mathrm{mod}\ r)|$, for $1 \le i, j \le n$. (Thus $a_{ij}$ are rectilinear distances on a rectangular $r \times s$ grid whose knot labels increase from the left to the right and from the bottom to the top.)
    Set $b_{ij}^{(0)} = w$, $g_{ij} = 2 - a_{ij}$, for $1 \le i, j \le n$.

While  While $g_{ij} \le 0$ for some $1 \le i, j \le n$ do

    Choose  Choose pair $(l, m)$ such that $a_{lm} = \max_{(ij): g_{ij} \le 0} \{a_{ij}\}$.
    Random  Randomly select a point $k_0$ among indices $k$ fulfilling the two following conditions[2]:
        1) $\min\{\lfloor l/r\rfloor, \lfloor m/r\rfloor\} \le \lfloor k/r\rfloor \le \max\{\lfloor l/r\rfloor, \lfloor m/r\rfloor\}$
        2) $\min\{l(\bmod r), m(\bmod r)\} \le k(\bmod r) \le \max\{l(\bmod r), m(\bmod r)\}$.
        Randomly choose $\Delta \in [0, z]$.
    Update  Set $b^{(0)}_{lm} := \Delta$, $b^{(0)}_{lk} := b^{(0)}_{lk} + (w-\Delta)$, $b^{(0)}_{mk} := b^{(0)}_{mk} + (w-\Delta)$ and $g_{lm} = g_{mk} = g_{lk} := 1$.
Endwhile

Permute  Set $b_{ij} = b^{(0)}_{\pi(i)\pi(j)}$, for $1 \le i, j \le n$.

End  Output matrices $A$ and $B$ and the optimal value $w\sum_{i=1}^n\sum_{j=1}^n a_{ij}$.
```

Next we prove the correctness of the Palubeckis' generator.

Theorem 2.9 (Palubeckis [138], 1988)
Let A, B be two $n \times n$ matrices generated by Algorithm 2.1, where n is part of the input. Then, the input permutation π is an optimal solution to QAP(A,B) with optimal value equal to $w\sum_{i=1}^n\sum_{j=1}^n a_{ij}$.

Proof. The proof is basically performed by induction on the number of repetitions of the while loop. Indeed, when the first while loop is executed, matrix $B^{(0)}$ is a constant matrix and therefore $QAP(A, B^{(0)})$ is a constant QAP, i.e., $Z(A, B^{(0)}, \phi) = w\sum_{i=1}^n\sum_{j=1}^n a_{ij}$, for each $\phi \in \mathcal{S}_n$. Thus, permutation id is an optimal solution to $QAP(A, B^{(0)})$.
Assume that the identical permutation id is an optimal solution to $QAP(A, B^{(i)})$ with optimal value $w\sum_{i=1}^n\sum_{j=1}^n a_{ij}$, where $B^{(i)}$ equals matrix $B^{(0)}$ after the ith iteration. We show that id is also an optimal permutation to $QAP(A, B^{(i+1)})$, where $B^{(i+1)}$ equals matrix $B^{(0)}$ after the (i+1)th iteration. $B^{(i+1)}$ is obtained from $B^{(i)}$ by applying "Update". That is, $B^{(i+1)} = B^{(i)} + \widehat{B}$, where $\widehat{B} = (\widehat{b}_{ij})$ is given by:

$$\widehat{b}_{ij} = \begin{cases} w-\Delta & \text{if } i = l \text{ and } j = k \\ w-\Delta & \text{if } i = m \text{ and } j = k \\ \Delta - w & \text{if } i = l \text{ and } j = m \\ 0 & \text{otherwise} \end{cases}$$

An elementary calculation reveals that $Z(A, \widehat{B}, id) = 0$. The last equality implies that id is an optimal solution to $QAP(A, \widehat{B})$. Considering this fact and the equality

$$Z(A, B^{(i+1)}, \phi) = Z(A, B^{(i)}, \phi) + Z(A, \widehat{B}, \phi),$$

[2]In a pictorial setting, an index k fulfilling these conditions is the label of a random point chosen on the (unique) smallest subgrid having two extremal vertices labeled by i and j.

which holds for each $\phi \in \mathcal{S}_n$, it is easily seen that *id* is an optimal solution to $QAP(A, B^{(i+1)})$ with optimal value $w \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}$. Assuming that at the end of the algorithm the while loop has been applied ℓ times, we have $B = B^{(\ell)^\pi}$. The claim follows immediately, when considering that equalities $Z(A, B, \phi) = Z(A, B^{(\ell)^\pi}, \phi) = Z(A, B^{(\ell)}, \phi \circ \pi^{-1})$ hold, for each $\phi \in \mathcal{S}_n$. ■

Once a test problem generator is proposed, a natural question arises, concerning the complexity of problems it produces. If these instances are "easy" in some sense, they are not really appropriate to be used for heuristics evaluation, as they are not representative for the more complex instances of the considered problem. The following theorem shows that the QAP instances generated by Palubeckis' generator are "easy", in the sense that their optimal value can be computed in polynomial time by solving an auxiliary linear program.

Theorem 2.10 (Li and Pardalos [117],1992)
Consider QAP(A,B), where A and B are matrices produced by Algorithm 2.1 under an unknown input. Consider all ordered triples of the form $\{(l,m),(l,k),(m,k)\}$, *where* l,m,k *are pairwise distinct indices ,* $1 \le l,m,k \le n$. *Fix an arbitrary order on the set of these triples and denote by* T_p *the* p^{th} *triple in this order. If* $T_p = \{(l,m),(l,k),(m,k)\}$, *then let* $T_p^+ = \{(l,k),(m,k)\}$ *and* $T_p^- = \{(l,m)\}$, *for all* p. *Then, the optimal value of QAP(A,B) is equal to* $z \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}$, *where* z *is the optimal value of the following linear program*

$$
\begin{aligned}
& z = \max w \\
& \textit{subject to} \\
& w + \sum_{(ij) \in T_p^-} \Delta_p - \sum_{(i,j) \in T_p^+} \Delta_p = b_{ij} \qquad 1 \le i,j \le n \\
& \Delta_p > 0 \qquad p = 1,2,\ldots,3\binom{n}{3}
\end{aligned}
$$

■

It is worthy to notice that nothing is known about the computational complexity of QAP instances generated by Palubeckis' generator. We believe that *finding an optimal solution* to these QAPs is NP-hard, although the corresponding decision problem is polynomially solvable.

Finally, notice that instead of using the distance matrix of a grid graph, one can start Palubeckis' generator with A being an arbitrary Euclidean matrix.

Li and Pardalos' generator for QAPs with a known optimal solution

Li and Pardalos propose another generator of QAP instances with a known optimal solution, called LP generator. This generator shares the basic underlying idea of Palubeckis' generator. It start with a trivial QAP instance with a known optimal solution and changes its coefficient matrices, so that the resulting QAP instance shares the same optimal solution with the initial one, but it is not trivial any more. Here, the strategy for changing up the coefficient matrices is based Proposition 1.1.

Algorithm 2.2 *LP generator for QAP instances with a known optimal solution.*
Input: Three positive natural numbers n, Δ_A, Δ_B and a permutation $\pi \in \mathcal{S}_n$.
Output: Two $n \times n$ matrices A, B such that π is an optimal solution to QAP(A,B).

```
Initialize Input $\Delta_A$ and $\Delta_B$.
    Set $a_{ij} = \Delta_A$, for $i \neq j$, $1 \leq i,j \leq n$, and $a_{ii} = 0$, for $1 \leq i \leq n$.
    Generate entries $b_{ij}^{(0)}$, $i \neq j$, $1 \leq i,j \leq n$, using a random generator with
    uniform distribution on $[0, \Delta_B]$. Set $b_{ii}^{(0)} = 0$, for $1 \leq i \leq n$.

Sort Sort $b_{ij}^{(0)}$, $i \neq j$, $1 \leq i,j \leq n$, increasingly and store the corresponding ranks in
    $r_{ij}$, respectively.

Random Randomly generate numbers $x_i$, for $i = 1, 2, \ldots, n(n-1)$, and sort them
    increasingly.

Update For $i \neq j$, $1 \leq i,j \leq n$, set $a_{ij} := a_{ij} - x_{r_{ij}}$.

Permute Set $b_{ij} = b_{\pi(i)\pi(j)}^{(0)}$, for $1 \leq i,j \leq n$.

End Output matrices $A$ and $B$.
```

The following theorem, whose simple proof is not given here, states the correctness of Algorithm 2.2.

Theorem 2.11 (Li and Pardalos [117], 1992)
For an input natural number n, an input permutation π and two input constants Δ_A and Δ_B, Algorithm 2.2 outputs two $n \times n$ matrices A, B such that π is an optimal solution to QAP(A,B). This is done in $O(n^2 \log n)$ time, assuming constant time for generating a random number. ■

Actually, a more general scheme for generating QAP instances with known optimal solution is described in [117]. This scheme makes use of the so called *sign-subgraphs* of a given graph. Applying this scheme with different sign-subgraph yields different QAP generators. In this context, the LP generator is obtained when the sign-subgraphs are as simple as possible: each of them consists of a single edge. It is interesting and surprising that QAP instances generated by applying the general scheme with more complex sign-subgraphs such as triangles and spanning trees are "easier" than those generated by the LP generator. Here, a QAP instance is considered to be "easy", if most of the heuristics applied to it find a solution near to the optimal solution in a relatively short time. As another interesting but not surprising result, it could be mentioned that instances of grid QAPs seem to be easier to solve than instances of symmetric QAPs and the latter seem to be easier to solve than instances of asymmetric QAPs, with respect to the quality of solutions produced by heuristics. On the other side, when considering the running times needed to find good solutions, the above order of simplicity seems to be reversed. An intuitive explanation for this behavior would be the large number of equal entries in symmetric and grid matrices. This implies the existence of large groups of feasible solutions with the same objective function value which obviously confuses matters for most heuristics.

Notice that Theorem 2.10 does not apply to QAP instances generated by the LP generator. The proof of this theorem, as well as Palubeckis' generator itself, essentially exploits the property of matrix A being Euclidean, whereas none of the two matrices generated by Algorithm 2.2 needs to be Euclidean.

In Chapter 5 we will see how to exploit the LP generator for generating instances of BiQAP with known optimal solution.

2.7 Asymptotic behavior

In this section some results on the asymptotic behavior of QAPs are presented. Among others, it is shown that if certain probabilistic conditions on the coefficient matrices of QAP(A,B) are fulfilled, the ratio between its "best" and "worst" objective function values almost surely approaches 1, as the size of the problem approaches infinity. As most of the known results on QAPs confirm the general belief on the extreme difficulty of this problem, this kind of asymptotic behavior is astonishing. Actually, it suggests that the relative error of every heuristic method almost surely vanishes as the size of the problem tends to infinity, i.e., every heuristic finds almost always an optimal solution when applied to large enough QAP instances. In other words, QAP becomes a kind of trivial as the size of the problem increases. Burkard and Fincke [30] identify a common combinatorial property of a number of problems which, under natural probabilistic conditions on data, behave as described above. This property seems to be also the key for the specific asymptotic behavior of the QAP. In the coming chapters, this general result of Burkard and Fincke and a stronger result of Szpankowski [173] is used for proving a similar asymptotic behavior of the BiQAP and of the CAP.

Next, we introduce some general results on the asymptotic behavior of some combinatorial optimization problems. Then we present some of the most important results on the asymptotic behavior of the QAP.

2.7.1 The astonishing asymptotic behavior of some combinatorial optimization problems

Consider a sequence $\mathrm{P_n}$, $n \in \mathbb{N}$, of (minimization) combinatorial optimization problems with sum objective function, as defined in Definition 1.3. Let $\mathcal{E}_n$, $\mathcal{F}_n$ be the ground set and the feasible solutions set of problem $\mathrm{P_n}$, $n \in \mathbb{N}$, respectively. Moreover, let $f_n\colon \mathcal{E}_n \to \mathbb{R}^+$ be the nonnegative cost function of problem $\mathrm{P_n}$. For $n \in \mathbb{N}$, a *worst solution* $X_{wor} \in \mathcal{F}_n$ is defined as follows:

$$\sum_{x \in X_{wor}} f_n(x) = \max_{X \in \mathcal{F}} \sum_{x \in X} f_n(x)$$

We will consider the ratio between the objective function values corresponding to an optimal (or best) and a worst solution, respectively. In [30], Burkard and Fincke show that this ratio is strongly related to the ratio between the cardinality of the set of the feasible solutions and the cardinality of an arbitrary feasible solution, under the assumption that all feasible solutions have the same cardinality. Then we get the following theorem.

Theorem 2.12 (Burkard and Fincke [30], 1985)
Let $f_n(x)$, $x \in \mathcal{E}_n, n \in \mathbb{N}$, be random variables identically distributed on $[0,1]$ with expected value $E := E(f_n(x))$ and variance $\sigma^2 = \sigma^2(f_n(x)) > 0$. Assume that for each $n \in \mathbb{N}$ all feasible solution $X \in \mathcal{F}_n$ have the same cardinality, say $|X_n|$. For a given $\epsilon > 0$ let ϵ_0 fulfill

$$0 < \epsilon_0 < \sigma^2 \textit{ and } 0 < \frac{E+\epsilon_0}{E-\epsilon_0} \le 1+\epsilon. \tag{2.21}$$

Furthermore, let the following two conditions be satisfied:

$$f_n(x), x \in X, \textit{ are pairwise independently distributed for every } X \in \mathcal{F}_n,\ n \in \mathbb{N}\ , \tag{2.22}$$

and

$$\lambda_0|X_n| - \ln|\mathcal{F}_n| \to \infty \quad \textit{as} \quad n \to \infty. \tag{2.23}$$

where λ_0 is defined by $\lambda_0 := 2(\epsilon_0\sigma/(\epsilon_0 + 2\sigma^2))^2$.
Then

$$P\left\{ \frac{\max\limits_{X\in\mathcal{F}_n} \sum\limits_{x\in X} f_n(x)}{\min\limits_{X\in\mathcal{F}_n} \sum\limits_{x\in X} f_n(x)} < 1+\epsilon \right\} \ge 1 - 2|\mathcal{F}_n|\exp(-|X_n|\lambda_0) \to 1 \textit{ as } n \to \infty$$

Proof. The proof of Burkard and Fincke [30] can be adapted to the case that not all feasible solutions have the same size, in the following way. First it has to be shown that

$$P\left\{ \exists X \in \mathcal{F}_n \ :\ \left|\sum_{x\in X}(f_n(x) - E)\right| \ge \epsilon_0|X| \right\} \le 2|\mathcal{F}_n|\exp(-|X_n|\lambda_0) \to 0 \tag{2.24}$$

as $n \to \infty$. Let us consider the following chain of inequalities:

$$\begin{aligned} & P\left\{\exists X \in \mathcal{F}_n \;:\; \left|\sum_{x\in X}(f_n(x)-E)\right| \geq \epsilon_0|X|\right\} \leq \\ \leq\; & \sum_{X\in\mathcal{F}_n} P\left\{\left|\sum_{x\in X}(f_n(x)-E)\right| \geq \epsilon_0|X|\right\} \leq \\ \leq\; & |\mathcal{F}_n|\, P\left\{\left|\sum_{x\in\widehat{X}}(f_n(x)-E)\right| \geq \frac{\epsilon_0\sqrt{|\widehat{X}|}}{\sigma}\sqrt{|\widehat{X}|}\sigma\right\} =: P_1 \end{aligned}$$

where $\widehat{X}$ is the feasible solution for which the above considered probability is maximum. Now a lemma of Rényi [153] can be applied with

$$D = \sqrt{\sum_{e\in\widehat{X}} \sigma^2(f_n(x))} = \sqrt{|\widehat{X}|}\sigma,\; K = 1\ ,$$

by defining

$$\mu := \frac{\epsilon_0\sqrt{|\widehat{X}|}}{\sigma}.$$

(Due to assumption (2.21) μ is defined as required in Rényi's lemma.) Thus we get

$$\begin{aligned} P_1 \;&\leq\; 2|\mathcal{F}_n|\exp\left(-\left(\frac{\epsilon_0\sqrt{|\widehat{X}|}}{\sigma}\right)^2 \Big/ 2\left(1+\frac{\epsilon_0\sqrt{|\widehat{X}|}}{\sigma}\cdot\frac{1}{2\sqrt{|\widehat{X}|}\sigma}\right)^2\right) = \\ &=\; 2|\mathcal{F}_n|\exp(-\lambda_0|\widehat{X}|) =: P_2 \end{aligned}$$

by exploiting the definition of λ_0. Applying (2.23) leads to $P_2 \to 0$, which proves relation (2.24). That relation is equivalent to

$$P\left\{\forall X \in \mathcal{F}_n : \left|\sum_{x\in X}(f_n(x)-E)\right| < \epsilon_0|X|\right\} \geq 1 - 2|\mathcal{F}_n|exp(-|X_n|\lambda_0) \to 1$$

as $n \to \infty$. Thus we obtain from (2.21)

$$\frac{\max\limits_{X\in\mathcal{F}_n}\sum\limits_{x\in X} f_n(x)}{\min\limits_{X\in\mathcal{F}_n}\sum\limits_{x\in X} f_n(x)} < \frac{\epsilon_0|X| + E|X|}{-\epsilon_0|X| + E|X|} \leq 1+\epsilon$$

with probability 1 as n approaches $+\infty$. ∎

Among the conditions of Theorem 2.12, the combinatorial condition represented by equation (2.23) means basically that the cardinality of the feasible solutions should be large enough with respect to the cardinality of the set of feasible solutions. Namely, the result of the theorem is true if the following equality holds:

$$\lim_{n\to\infty} \frac{\log|\mathcal{F}|}{|X_n|} = 0 \qquad (*)$$

The other conditions are natural probabilistic requirements to be put on the problems coefficients. Some traditional problems which fulfill condition $(*)$ are the quadratic assignment problem, minimum perfect matching problem and the shortest path problem, whereas the traveling salesman problem and the linear assignment problem do not fulfill it.

The result of Theorem 2.12 means that the ratio between the best and the worst value of the objective function approaches 1, *with probability tending to 1*, as the "size" of the problem approaches infinity. Thus, we have convergence *in probability.* Under one additional natural (combinatorial) assumption, Szpankowski strengthens this result and improves the range of the convergence to *almost surely.*

Theorem 2.13 (Szpankowski [173], 1995)
Let $\mathrm{P_n}$ *be a sequence of minimization combinatorial optimization problems with sum objective functions as above. Assume that the following conditions are fulfilled:*

(C1) *For all* $X \in \mathcal{F}_n$, $|X| = |X_n|$, *where* X_n *is a fixed feasible solution in* $\mathcal{F}_n$.

(C2) *The costs* $f(x)$, $x \in X$, $X \in \mathcal{F}_n$, $n \in \mathbb{N}$, *are random variables identically and pairwise independently distributed on* $[0,1]$. *The mean value* μ, *the variance and the third moment of the common distribution are finite.*

(C3) *The worst values of the objective function,* $\max_{X\in\mathcal{F}_n} \sum_{x\in X} f(x)$ *form a nondecreasing sequence with respect to* n.

(C4) $|\mathcal{F}_n|$ *and* $|X_n|$ *tend to* ∞ *as* n *tends to* ∞ *and moreover,* $\log|\mathcal{F}_n| = o(|X_n|)$.

Then, the following equalities hold almost surely:

$$\min_{X\in\mathcal{F}} \sum_{x\in X} f(x) = |X_n|\mu - o(|X_n|)\ , \quad \max_{X\in\mathcal{F}} \sum_{x\in X} f(x) = |X_n|\mu + o(|X_n|) \qquad \blacksquare$$

2.7.2 On the asymptotic behavior of the QAP

Theorems 2.12 and 2.13 can be applied to QAP instances whose coefficients fulfill some natural probabilistic conditions. The main point is that the QAP fulfills the combinatorial condition (C4) in Theorem 2.13 (or equivalently the condition represented by (2.23) in Theorem 2.12). Indeed, when considering a QAP(A,B) of size n in the frame of Definition 1.3, as in Section 1.3.6, we have $|\mathcal{F}_n| = n!$ and $|X| = n^2$, for each $X \in \mathcal{F}_n$. By applying Stirling's formula it is easy to see that $\lim\limits_{n\to\infty} \frac{\log(n!)}{n^2} = 0$. So we immediately get the following corollary:

Corollary 2.14 *Consider a sequence of problems $QAP(A^{(n)}, B^{(n)})$, $n \in \mathbb{N}$, with $n \times n$ coefficient matrices $A^{(n)} = \left(a_{ij}^{(n)}\right)$ and $B = \left(b_{ij}^{(n)}\right)$. Assume that $a_{ij}^{(n)}$ and $b_{ij}^{(n)}$, $n \in \mathbb{N}$, $1 \le i, j \le n$, are pairwise independently distributed random variables on $[0, M]$, where M is a positive constant. Moreover, let all of these variables have finite mean, variance and third moment. Furthermore, assume that $a_{ij}^{(n)}$, $n \in \mathbb{N}$, $1 \le i, j \le n$, have the same distribution and $b_{ij}^{(n)}$, $n \in \mathbb{N}$, $1 \le i, j \le n$, have also the same distribution (eventually different from that of $a_{ij}^{(n)}$). Let us denote by $\pi_{opt}^{(n)}$ and $\pi_{wor}^{(n)}$ an optimal and a worst solution of $QAP(A^{(n)}, B^{(n)})$ respectively, i.e.,*

$$Z\left(A^{(n)}, B^{(n)}, \pi_{opt}^{(n)}\right) = \min_{\pi \in \mathcal{S}_n} Z\left(A^{(n)}, B^{(n)}, \pi\right)$$

$$Z\left(A^{(n)}, B^{(n)}, \pi_{wor}^{(n)}\right) = \max_{\pi \in \mathcal{S}_n} Z\left(A^{(n)}, B^{(n)}, \pi\right)$$

Then the following equality holds almost surely:

$$\lim_{n\to\infty} \frac{Z\left(A^{(n)}, B^{(n)}, \pi_{opt}^{(n)}\right)}{Z\left(A^{(n)}, B^{(n)}, \pi_{wor}^{(n)}\right)} = 1 \qquad \blacksquare$$

The above result suggests that the objective function values of $QAP(A^{(n)}, B^{(n)})$ (corresponding to arbitrary feasible solutions) get somehow close to their expected value $n^2E(A)E(B)$, as the size of the problem increases. Here, $E(A)$ and $E(B)$ are the expected values of $a_{ij}^{(n)}$ and $b_{ij}^{(n)}$, $n \in \mathbb{N}$, $1 \le i, j \le n$, respectively. Several authors provide different analytical evaluations of this "getting close", under different probabilistic conditions. We should mention here the works of Frenk, Houweninge and Rinnooy Kan [62] and the papers of Rhee [155, 156]. The following theorem states two important results proved in [62] and [156], respectively.

Theorem 2.15 *(Frenk et al. [62], 1986, Rhee [156], 1991)*
Consider the sequence of $QAP(A^{(n)}, B^{(n)})$ as in Corollary 2.14. Assume that the following conditions are fulfilled:

(C1) *$a_{ij}^{(n)}$, $b_{ij}^{(n)}$, $n \in \mathbb{N}$, $1 \le i, j \le n$, are random variables pairwise independently distributed on $[0, M]$.*

(C2) *$a_{ij}^{(n)}$, $n \in \mathbb{N}$, $1 \le i, j \le n$, have the same distribution on $[0, M]$. $b_{ij}^{(n)}$, $n \in \mathbb{N}$, $1 \le i, j \le n$, have also the same distribution on $[0, M]$ (eventually different from that of $a_{ij}^{(n)}$).*

Let $E(A)$, $E(B)$ be the expected values of variables $a_{ij}^{(n)}$ and $b_{ij}^{(n)}$, respectively. Then, there exist a constant K_1, independent on the size of the problem, such that the following inequality holds almost surely, for $\pi \in \mathcal{S}_n$, $n \in \mathbb{N}$

$$\limsup_{n\to\infty} \frac{\sqrt{n}}{\sqrt{\log n}} \left| \frac{Z(A^{(n)}, B^{(n)}, \pi)}{n^2 E(A)E(B)} - 1 \right| \le K_1$$

Moreover, let Y be a random variable defined by

$$Y = Z\left(A^{(n)}, B^{(n)}, \pi_{opt}^{(n)}\right) - n^2 E(A)E(B)\,,$$

where $\pi_{opt}^{(n)}$ is an optimal solution to $QAP(A^{(n)}, B^{(n)})$. Then, there exists another constant K_2, also independent on the size of the problem, such that

$$\frac{1}{K_2} n^{3/2} (\log n)^{1/2} \le E(Y) \le K_2 n^{3/2} (\log n)^{1/2}$$

$$P\left\{|Y - E(Y)| \ge t\right\} \le 2\exp\left(\frac{-t^2}{4n^2 \|a\|_\infty^2 \|b\|_\infty^2}\right)$$

where $E(Y)$ denotes the expected value of variable Y. ■

Part II

Easy and Hard Cases of the QAP: Drawing a Borderline

Chapter 3

QAPs on Specially Structured Matrices

In this chapter and in the next one we consider the challenging question of identifying polynomially solvable cases of the QAP. As there is no hope to find a polynomial time algorithm for solving the general QAP and as QAP instances arising in different practical applications may often have a special structure, it is interesting to derive polynomial time algorithms for solving special cases of the problem. Moreover, any information on provably difficult ($\mathcal{NP}$-hard) cases of the problem is also of relevance. Additionally, we formulate a set of open questions on QAPs with unknown complexity, which we consider to be a promising object of further research in this direction.

A *special case* or a *restricted version of the QAP* is the problem class consisting of all QAP instances whose coefficient matrices fulfill certain prespecified properties. In this chapter we present results on versions of the QAP whose coefficient matrices have specific structural combinatorial properties, referring to certain matrix classes such as Monge and Anti-Monge matrices, Toeplitz and circulant matrices, sum and product matrices, graded matrices etc. Among polynomially solvable restricted versions of QAPs two notable groups can be distinguished: *constant QAPs* and *constant permutation QAPs*. The *constant QAPs* are problems whose objective function value does not depend on the feasible solution, i.e., all feasible solutions yield the same value of the objective function. Hence, in this case each feasible solution is an optimal solution and the problem is kind of trivial.

The *constant permutation QAPs* are defined by the following property. For each problem instance there is an optimal solution which depends only on the structure of the instance and on the instance size, but not on the entries of its coefficient matrices. For such QAPs, there is usually a rule telling us how to construct the above mentioned optimal solution. The permutation constructed according to this rule is called *constant permutation* with respect to the considered QAP version. Obviously, the constant permutation need not be the unique optimal solution to all instances of the constant permutation QAP. The terms *constant QAP* and *constant permutation*

QAP maintain their meaning throughout this part of the thesis.

The rest of this chapter is organized as follows. In the first section we introduce the matrix classes which we are dealing with and related notations. In the second section, we continue by summarizing some preliminary results and elementary observations. The third section lists some simple polynomially solvable cases of QAP. Then, we investigate QAPs on *Monge* and *Monge-like* matrices in Section 3.4. In Section 3.5 QAPs with *circulant* and *limited bandwidth* coefficient matrices are considered. Results on the so called *taxonomy problem* are included in this section. It follows a detailed analysis of a restricted version of QAPs with a Toeplitz and an Anti-Monge matrix in Section 3.6. Besides a number of results on special cases of this QAP version, we also present three interesting applications of it in Section 3.7. Among these applications two are of practical nature, the so called *turbine problem* and a *data arrangement problem*. The third application concerns the *traveling salesman problem* on symmetric Monge matrices. Our main theorem of Section 3.6 unifies and generalizes some well known results on these three problems. Concluding, Section 3.8 overviews the achieved results and states some next-to-solve open problems.

3.1 Definitions

In this section we define the classes of matrices we are going to deal with and introduce the corresponding notations. It will turn out that special versions of QAPs with coefficient matrices having Monge and Monge like properties are interesting. Let us first recall the definition of Monge and Anti-Monge[1] matrices as they usually occur in the literature.

Definition 3.1 *A matrix $A = (a_{ij})$ is called a* Monge matrix *if its elements satisfy $a_{ij} + a_{rs} \le a_{is} + a_{rj}$, for $1 \le i < r \le n$ and $1 \le j < s \le n$. The class of Monge matrices is denoted by* MONGE.

A matrix $A = (a_{ij})$ is called an Anti-Monge matrix *if its elements satisfy $a_{ij} + a_{rs} \ge a_{is} + a_{rj}$, for $1 \le i < r \le n$ and $1 \le j < s \le n$. The class of Anti-Monge matrices is denoted by* ANTI-MONGE.

Other interesting classes of matrices with simple combinatorial structure, which lead to well solvable versions of QAP, are defined below.

Definition 3.2 *A matrix $A = (a_{ij})$ is called a* sum matrix *(*product matrix*) if there exist real numbers α_i^r and α_i^c, $1 \le i \le n$, such that $a_{ij} = \alpha_i^r + \alpha_j^c$ ($a_{ij} = \alpha_i^r \alpha_j^c$), for $1 \le i, j \le n$. The classes of sum and product matrices are denoted by* SUM *and*

[1]Different authors use different names for Anti-Monge matrices. Some frequent alternative names are "inverse Monge" and "contra Monge".

PRODUCT *respectively.* (α_i^r) *and* (α_i^c) *are called* row and column generating vectors *of the corresponding matrix, respectively.*

A matrix $A = (a_{ij})$ *is called a* small matrix (large matrix) *if there exist real numbers* α_i^r *and* α_i^c, $1 \le i \le n$, *such that* $a_{ij} = \min(\alpha_i^r, \alpha_j^c)$ $(a_{ij} = \max(\alpha_i^r, \alpha_j^c))$, *for* $1 \le i, j \le n$. *The classes of small and large matrices are denoted by* SMALL *and* LARGE, *respectively. Again,* (α_i^r) *and* (α_i^c) *are termed* row and column generating vectors *of the corresponding matrix, respectively.*

The matrix $A = (a_{ij})$ *with* $a_{ij} = (-1)^{i+j}$, $1 \le i, j \le n$, *is called a* chessboard matrix. *The class of chessboard matrices is denoted by* CHESS.

For the so called graded matrices we adopt the definition used in [182].

Definition 3.3 *A matrix A is* graded on its rows (graded on its columns) *if all its rows (columns) have the same monotonicity. i.e., either all of them are nondecreasing or all of them are nonincreasing.*

A matrix $A = (a_{ij})$ *is called* left-higher graded *if all its rows and columns are nondecreasing, i.e.,* $a_{ij} \le a_{i,j+1}$, $1 \le j \le n-1$, $1 \le i \le n$, *and* $a_{ij} \le a_{i+1,j}$, $1 \le i \le n-1$, $1 \le j \le n$. *The matrix A is called* right-lower graded *if both its rows and its columns are nonincreasing, i.e.,* $a_{ij} \ge a_{i,j+1}$, $1 \le j \le n-1$, $1 \le i \le n$, *and* $a_{ij} \ge a_{i+1,j}$, $1 \le i \le n-1$, $1 \le j \le n$. *In an analogous way, the terms* left-lower graded *and* right-higher graded *are introduced. The above defined properties are abbreviated as* LHG, RLG, LLG *and* RHG *respectively. In a pictorial setting, the two first letters of these abbreviations describe the position of the smallest entry in the matrix.*

It will turn out that some versions of QAPs with diagonally structured matrices are easy to solve. The following definition introduces some interesting classes of diagonally structured matrices.

Definition 3.4 *A matrix* $A = (a_{ij})$ *is a* Toeplitz matrix, *if there exist* $2n-1$ *real numbers* $c_{-n+1}, \ldots, c_{n-1}$ *such that* $a_{ij} = c_{i-j}$, *for all* $1 \le i, j \le n$. *The class of Toeplitz matrices is denoted by* TOEPLITZ.

If the numbers c_i *in the definition of a Toeplitz matrix A fulfill the equalities* $c_k = c_{k \,(mod\, n)}$, *for all* $-n+1 \le k \le n-1$, *then A is called a* circulant matrix. *This class is denoted by* CIRCULANT.

Consider matrix $A = (a_{ij})$. *In the case that d is the smallest integer having the property* $a_{ij} = 0$ *for all* $1 \le i, j \le n$ *with* $|i-j| \ge d$, *then matrix A is called a* bandwidth-d matrix. *The class of bandwidth-d matrices is denoted* BANDWIDTH-*d*.

Additionally, we use the following definition which is not frequent in the literature: An $n \times n$ matrix A is called an *odd matrix* if n is an odd integer and is called an *even matrix* if n is even. The abbreviations SYM, SKEW, ODD and EVEN are used to denote the properties symmetric, skew symmetric, odd and even, respectively, where symmetric and skew symmetric matrices are defined in the first chapter.

Through the rest of the paper, we use the notation Class(Prop1, Prop2, Prop3) for the subclass of matrices in Class with properties Prop1, Prop2 and Prop3. For example, the notation Monge(Sym,lhG,Odd) will be used to denote the class of symmetric Monge matrices of odd size with nondecreasing rows and columns.

For a class Class of matrices, we denote by PermClass the class consisting of matrices $A^\pi = (a_{\pi(i)\pi(j)})$, for all $A \in$ Class and for all $\pi \in \mathcal{S}_n$. Obviously, for two matrix classes Class1 and Class2 the inclusions PermClass1$\subseteq$PermClass2 and Class1$\subseteq$PermClass2 are equivalent. If these inclusions hold, we write Class1$\subseteq_\pi$Class2. For example, it is easily seen that Product(Sym) $\subseteq_\pi$lhG. The case where both inclusions Class1$\subseteq_\pi$Class2 and Class2$\subseteq_\pi$Class1 simultaneously hold will be abbreviated by Class1$=_\pi$Class2. As an example, note that lhG$=_\pi$rlG and rhG$=_\pi$llG. The significance of these notations becomes clear when considering that problems QAP(A,B) and QAP(A_1,B_1) with $A \in$Class1, $B \in$ Class2 and $A_1 \subseteq_\pi$Class1, $B_1 \subseteq_\pi$Class2 are equivalent. From now on, the sentence "problems *QAP(A,B) and QAP(A_1, B_1) are equivalent*" means that once we know an optimal solution to one of these problems, an optimal solution to the other one can be constructed in polynomial time.

The class of problems QAP(A,B) with matrices $A \in$ Class1 and $B \in$ Class2 is denoted by Class1$\times$Class2. For a class Class of matrices, we denote by NegClass the class consisting of matrices $-A$, for all $A \in$Class. As an illustrative example consider the equality Anti-Monge=NegMonge. Finally, the class of all matrices is denoted by Matrix.

We complete this section by introducing a relaxation of the QAP, where the rows and columns of the coefficient matrix A may be permuted by two independent permutations $\pi, \varphi \in S_n$. Namely,

$$\min_{\varphi,\pi\in S_n} \sum_{i=1}^{n}\sum_{j=1}^{n} a_{\varphi(i)\pi(j)} b_{ij} \tag{3.1}$$

We denote this problem by RQAP. The notations RQAP(A,B), $Z(A,B,\varphi,\pi)$, *optimal value* and *optimal solution* are defined similarly as for the QAP. Clearly, RQAP is a relaxation of the QAP, i.e. the following inequality holds:

$$\min_{\pi\in S_n} \sum_{i=1}^{n}\sum_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij} \geq \min_{\varphi,\pi\in S_n} \sum_{i=1}^{n}\sum_{j=1}^{n} a_{\varphi(i)\pi(j)} b_{ij} \tag{3.2}$$

That is, the optimal value of QAP(A,B) is larger than or equal to the optimal value of RQAP(A,B). Therefore, if a pair of permutations (π, π) is an optimal solution to RQAP(A,B), then π is an optimal solution to QAP(A,B). This relationship between RQAP(A,B) and QAP(A,B) is exploited in several proofs in this chapter.

3.2 Preliminaries and elementary observations

In this section we introduce some preliminary and elementary observations to be used through the current and the next chapter. First, we formulate as an observation the following easily seen fact:

Observation 3.1 *Let the matrices A and B be linear combinations of arbitrarily given matrices A_i, B_i, $1 \le i \le k$, and constants μ_i, υ_i, $1 \le i \le k$, i.e. $A = \sum_{i=1}^k \mu_i A_i$ and $B = \sum_{i=1}^k \upsilon_i B_i$. Then*

$$Z(A, B, \pi) = \sum_{i=1}^{k} \sum_{j=1}^{k} \mu_i \upsilon_j Z(A_i, B_j, \pi) .$$

■

Next, we present some results on the combinatorial structure of the matrix classes MONGE and ANTI-MONGE. It can be shown that each of these matrix classes is a cone. Moreover, for each of the cones related to these matrix classes, 0-1 matrices generating the corresponding extremal rays can explicitly be specified. For more details about these results and their proofs the reader is referred to [162] and [27].

First, let us introduce four types of 0-1 matrices which generate the extremal rays of the cone of the nonnegative Monge matrices. Consider the 0-1 $n \times n$ matrices $H^{(p)} = \left(h_{ij}^{(p)}\right)$, $V_{ij}^{(q)} = \left(v_{ij}^{(q)}\right)$, $L^{(p,q)} = \left(l_{ij}^{(p,q)}\right)$ and $R^{(p,q)} = \left(r_{ij}^{(p,q)}\right)$, defined in the following way, for $1 \le p, q \le n$:

$$h_{ij}^{(p)} = \begin{cases} 1 & \text{if} \quad i = p \\ 0 & \text{otherwise} \end{cases} \qquad v_{ij}^{(q)} = \begin{cases} 1 & \text{if} \quad j = q \\ 0 & \text{otherwise} \end{cases}$$

$$l_{ij}^{(p,q)} = \begin{cases} 1 & \text{if} \quad i \ge n-p+1,\ j \le q \\ 0 & \text{otherwise} \end{cases} \qquad r_{ij}^{(p,q)} = \begin{cases} 1 & \text{if} \quad i \le p,\ j \ge n-q+1 \\ 0 & \text{otherwise} \end{cases}$$

Thus, matrix $H^{(p)}$ has a row of 1-entries, namely the row with index p, and the rest of its entries are equal to 0. Similarly, matrix $V^{(q)}$ has a column of 1-entries, namely the column with index q, and the rest of its entries equal 0. Matrix $L^{(p,q)}$ has a $p \times q$ submatrix with 1-entries in the left-lower corner; the rest of its entries equal 0. Similarly, matrix $R^{(p,q)}$ has a $p \times q$ submatrix with 1-entries in the right-higher corner; the rest of its entries are equal 0. Consider, for example, the 3×3 matrices $H^{(2)}$, $V^{(3)}$, $L^{(1,2)}$ and $R^{(2,2)}$ represented below:

$$H^{(2)} = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix} V^{(3)} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} L^{(1,2)} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} R^{(2,2)} = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

Now, the cone of the nonnegative Monge matrices can be fully described as follows:

Proposition 3.2 (Rudolf and Woeginger 1995, [162])
The nonnegative $n \times n$ Monge matrices form a cone. Its extremal rays are generated by the 0-1 matrices $H^{(i)}$, $V^{(i)}$, for $1 \le i \le n$, and $L^{(i,j)}$, $R^{(i,j)}$, for $1 \le i,j \le n$. ■

Sometimes we will restrict our investigations to symmetric Monge matrices. For the characterization of the cone of symmetric $n \times n$ Monge matrices, the matrices $S^{(p)} = H^{(p)} + V^{(p)}$, $1 \le p \le n$, and $T^{(p,q)} = L^{(p,q)} + R^{(q,p)}$, $1 \le p,q \le n$, $p+q \le n$, are of interest. For example, consider the 3×3 matrices $S^{(2)}$ and $T^{(1,2)}$ given below:

$$S^{(2)} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad T^{(1,2)} = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Proposition 3.3 (Rudolf and Woeginger 1995, [162])
The nonnegative symmetric $n \times n$ Monge matrices form a cone. Its extremal rays are generated by the matrices $S^{(i)}$, $1 \le i \le n$, and $T^{(i,j)}$, $1 \le i,j \le n$, $i+j \le n$. Thus, for every nonnegative $n \times n$ Monge matrix A, there exist nonnegative numbers κ_i, $1 \le i \le n$, and μ_{ij}, $1 \le i,j \le n$, $i+j \le n$, such that:

$$A = \sum_{i=1}^{n} \kappa_i S^{(i)} + \sum_{i=1}^{n-1} \sum_{j=1}^{n-i} \mu_{ij} T^{(i,j)}$$ ■

Some of our polynomiality results on QAPs with Monge or Anti-Monge coefficient matrices will additionally require the Monge-like coefficient matrix to be graded, i.e., that the entries in the columns and/or rows are monotone. Analogously to the nonnegative Monge matrices, the nonnegative left-higher graded Anti-Monge matrices and the nonnegative right-lower graded Monge matrices form also cones, respectively. The following simple observation helps to describe this structure.

Observation 3.4 *(a) A matrix A is Anti-Monge if and only if*

$$\Delta_{ij} := a_{ij} - a_{i,j-1} - a_{i-1,j} + a_{i-1,j-1} \ge 0, \text{ for } 1 < i,j \le n. \tag{3.3}$$

(b) An Anti-Monge matrix is left-higher graded if its first row and its first column are nondecreasing, i.e., if

$$\Delta_{i1} := a_{i1} - a_{i-1,1} \ge 0, \text{ for } 1 < i \le n, \text{ and} \tag{3.4}$$
$$\Delta_{1j} := a_{1j} - a_{1,j-1} \ge 0, \text{ for } 1 < j \le n. \tag{3.5}$$

(c) A left-higher graded Anti-Monge matrix is non-negative if

$$\Delta_{11} := a_{11} \ge 0. \tag{3.6}$$

(d) Moreover, a matrix A is completely determined by the n^2 values Δ_{ij}, $1 \leq i, j \leq n$. ■

Note that a matrix A is a non-negative left-higher graded Anti-Monge matrix if and only if the $(n+1) \times (n+1)$ matrix obtained by bordering A with an additional top row of zeros and an additional left column of zeros is an Anti-Monge matrix. In this way, inequalities (3.4)–(3.6) become special cases of (3.3), and the additional requirements of monotonicity and non-negativity appear quite natural for Anti-Monge matrices.

For the characterization of nonnegative matrices of classes MONGE(RLG) and ANTI-MONGE(LHG) we make use of the $0-1$ matrices $E^{(p,q)} = \left(e_{i,j}^{(p,q)}\right)$ and $C^{(p,q)} = \left(c_{ij}^{(p,q)}\right)$, for $1 \leq p, q \leq n$, defined as follows:

$$e_{i,j}^{(p,q)} = \begin{cases} 1 & \text{if} \quad i \leq p \text{ or } j \leq q \\ 0 & \text{otherwise} \end{cases}$$
$$c_{ij}^{(p,q)} = \begin{cases} 1 & \text{if} \quad i \geq n-p+1,\ j \geq n-q+1 \\ 0 & \text{otherwise} \end{cases}$$

Consider for example the 3×3 matrices $E^{(2,1)}$ and $C^{(2,1)}$ given below:

$$E^{(2,1)} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \end{pmatrix} \qquad C^{(2,1)} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Theorem 3.5 *The nonnegative left-higher graded $n \times n$ Anti-Monge matrices form a cone. The extremal rays of this cone are generated by the $0-1$ matrices $C^{(p,q)}$, $1 \leq p, q \leq n$.*

Proof. Since the nonnegative left-higher graded Anti-Monge matrices are defined by a homogeneous system of linear inequalities (3.3)–(3.6), they form a cone. Part (d) of Observation 3.4 implies that the mapping from the $n \times n$ matrices A to the $n \times n$ matrices $\Delta = (\Delta_{ij})$ is a one-to-one linear transformation. In the transformed "Δ-space" the n^2 defining inequalities of Observation 3.4 take a very simple form: $\Delta_{ij} \geq 0$ for all $1 \leq i, j \leq n$. Hence the extreme rays of the transformed cone are just the coordinate axes in Δ-space. A unit vector in Δ-space, i.e., a matrix with $\Delta_{pq} = 1$ for some pair (p, q) and $\Delta_{ij} = 0$ for all other pairs corresponds just to the matrix $C^{(n-p+1,n-q+1)}$ in the original space. This shows that the matrices $C^{(pq)}$, $1 \leq p, q \leq n$, generate the extremal rays of the cone. ■

From this theorem we get the following straightforward corollary, whose proof is omitted because of its simplicity.

Corollary 3.6 *The nonnegative right-lower graded $n \times n$ Monge matrices form a cone. The extremal rays of this cone are generated by the $0-1$ matrices $E^{(p,q)}$, $1 \leq p, q \leq n$.* ■

In the case that a problem at hand is believed to be a constant permutation QAP and moreover, there exists some pointer to its eventual constant permutation, then a special combinatorial structure of coefficient matrices as above can be of benefit. Namely, since the coefficient matrices of the considered QAP can be expressed as nonnegative linear combinations of 0-1 matrices generating the extremal rays of the corresponding cones, one can restrict the investigations to simpler QAPs with such 0-1 coefficient matrices. Thus, it is sufficient to prove that the conjectured constant permutation is indeed an optimal solution for these simple instances of QAP with 0-1 coefficient matrices. By applying Observation 3.1, this result extends then to the whole set of instances of the considered problem.

3.3 Simple polynomially solvable cases of the QAP

In this section we describe some simple polynomially solvable cases of the QAP. First let us recall Proposition 1.1 which is exploited in many proofs throughout this chapter: Given two vectors $U = (u_i)$ and $V = (v_i)$, the minimum of the scalar product of the permuted vectors $\langle U^{(\phi)}, V^{(\psi)} \rangle$ over all pairs of permutations (ϕ, ψ) is achieved when $U^{(\phi)}$ and $V^{(\psi)}$ are nonincreasing and nondecreasing, respectively. The following proposition, which is probably the very first result on well solvable cases of the QAP, is a straightforward corollary of Proposition 1.1:

Proposition 3.7 (Krushevski 1964, [108])
Consider two $n \times n$ matrices $A = (a_{ij})$ and $B = (b_{ij})$ such that for all 4-tuples of indices (i, j, k, l) the following equivalence holds:

$$(\ a_{ij} \geq a_{kl} \ ) \Longleftrightarrow (\ b_{ij} \leq b_{kl} \ )$$

Then, the identity permutation id is an optimal solution to QAP(A,B). ■

The following theorem states a generalization of Proposition 3.7:

Theorem 3.8 *The identity permutation id is an optimal solution to QAPs on the classes* LHG×RLG, LLG×RHG.

Proof. We prove the result only for QAPs on the class LHG×RLG. In the other case the proof is completely analogous.
Let $A = (a_{ij})$ be an $n \times n$ matrix with nondecreasing rows and columns and let $B = (b_{ij})$ be an $n \times n$ matrix with nonincreasing rows and columns. Consider the relaxation RQAP(A,B) and an arbitrary pair of permutations (φ, π) in $\mathcal{S}_n$. It is easy to see that the following inequalities hold, due to Proposition 1.1:

$$\begin{aligned} Z(A,B,\varphi,\pi) &= \sum_{i=1}^{n}\sum_{j=1}^{n} a_{\varphi(i)\varphi(j)}b_{ij} \\ &\geq \sum_{i=1}^{n}\sum_{j=1}^{n} a_{i\pi(j)}b_{ij} \geq \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}b_{ij} = Z(A,B,id,id) \end{aligned} \tag{3.7}$$

Inequalities (3.7) show that (id, id) is an optimal solution to RQAP(A,B). Hence, *id* is an optimal solution to QAP(A,B). ■

Notice that both Proposition 3.7 and Theorem 3.8 describe classes of constant permutation QAPs. At this point it is probably interesting to notice that the complexity of the apparently simple problem LHG×RHG remains an open question.

SKEW×SYM is another class of constant QAPs:

Theorem 3.9 *The problem* SKEW×SYM *is a constant QAP; each feasible solution yields an objective function value equal to* 0.

Proof. Let A be a skew symmetric $n \times n$ matrix and let B be a symmetric $n \times n$ matrix. Rewrite the objective function of QAP(A,B) as

$$Z(A,B,\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{\pi(i)\pi(j)}b_{ij} = \sum_{j=2}^{n}\sum_{i=1}^{j-1} a_{\pi(i)\pi(j)}b_{ij} + \sum_{i=1}^{n} a_{\pi(i)\pi(i)}b_{ii} + \sum_{i=2}^{n}\sum_{j=1}^{i-1} a_{\pi(i)\pi(j)}b_{ij} \ .$$

In the right hand side of the above equality, the first and the third term cancel each other due to symmetry conditions. Since the diagonal elements of a skew symmetric matrix are equal to 0, the remaining term $\sum_{i=1}^{n} a_{\pi(i)\pi(i)}b_{ii}$ equals 0 and this completes the proof. ■

The next theorem represents an easy-to-prove but surprising result:

Theorem 3.10 *The problem* SUM×MATRIX *is solvable in* $O(n^3)$ *time, where* n *is the size of the problem.*

Proof. The proof basically consists on transforming QAP(A,B) to a linear assignment problem, in the case that A is an $n\times n$ sum matrix and B is an arbitrary $n\times n$ matrix. Indeed, let (α_i^r) and (α_i^c) be the generating vectors of matrix A. The objective function of QAP(A,B) can be transformed as follows:

$$\begin{aligned} Z(A,B,\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{\pi(i)\pi(j)}b_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha^r_{\pi(i)} + \alpha^c_{\pi(j)})b_{i,j} = \sum_{i=1}^{n}\left(\alpha^r_{\pi(i)}\sum_{j=1}^{n} b_{i,j}\right) + \\ \sum_{j=1}^{n}\left(\alpha^c_{\pi(j)}\sum_{i=1}^{n} b_{i,j}\right) = \sum_{j=1}^{n}\alpha^r_{\pi(j)}\beta^r_j + \sum_{j=1}^{n}\alpha^c_{\pi(j)}\beta^c_j \ , \end{aligned} \tag{3.8}$$

where β_i^r, β_i^c, $1 \le i \le n$, are the sums of the i^{th} row and i^{th} column of matrix B, respectively. Now consider an $n \times n$ matrix $D = (d_{ij})$ defined as follows:

$$d_{ij} = \alpha_i^r \beta_j^r + \alpha_i^c \beta_j^c$$

With this notation we have $Z(A,B,\pi) = \sum_{i=1}^{n} d_{\pi(i)i}$. Hence, solving QAP(A,B) is equivalent to solving a linear assignment problem, namely $\min_{\pi \in \mathcal{S}_n} \sum_{i=1}^{n} d_{\pi(i)i}$. This completes the proof. ■

If matrices A and/or B in the above theorem have some "nice property" the problem Sum×Matrix becomes even "easier" to solve. Two results of this type are represented by the following theorem:

Theorem 3.11 *(a) The problem* Sum×Circulant *is a constant QAP, i.e., every permutation $\pi \in \mathcal{S}_n$ is an optimal solutions to* Sum×Circulant *of size n.*

(b) If at least one of the matrices A, B is symmetric or skew symmetric, then QAP(A,B) with $A \in$ Sum *is solvable in $O(n^2)$ time, where n is the size of the problem.*

Proof.

Proof of (a). Consider QAP(A,B) where $A = (a_{ij})$ and $B = (b_{ij})$ are two $n \times n$ matrices, with $A \in$ Sum and $B \in$ Circulant. For an arbitrary $\pi \in \mathcal{S}_n$, we have:

$$Z(A,B,\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij} = \sum_{p=0}^{n-1} \left(c_p \sum_{i=1}^{n} a_{\pi(i)\pi(k_{p+i})} \right), \tag{3.9}$$

where $c_0, c_1, \ldots, c_{n-1}$ are real numbers as introduced in the definition of circulant matrices, and $k_i := i \ (mod\ n)$, for $i \in \{1, 2 \ldots, 2n-1\} \setminus \{n\}$ and $k_n := n$. Let (α_i^r) and (α_i^c) be the generating vectors of sum matrix A. Then, the righthand side of equality (3.9) can be written as follows:

$$Z(A,B,\pi) = \sum_{p=0}^{n-1} \left(c_p \sum_{i=1}^{n} \alpha_{\pi(i)}^r \right) + \sum_{p=0}^{n-1} \left(c_p \sum_{i=1}^{n} \alpha_{\pi(k_{p+i})}^c \right) =$$

$$\sum_{p=0}^{n-1} \left(c_p \sum_{i=1}^{n} \alpha_i^r \right) + \sum_{p=0}^{n-1} \left(c_p \sum_{j=1}^{n} \alpha_j^c \right) .$$

Note that the right hand side of the last equality does not depend on permutation π. This completes the proof in this case.

Proof of (b). Consider QAP(A,B) with $A \in$ Sum and $B \in$ Matrix(Skew). (The three remaining cases can be discussed analogously.) Denote by β_i^r and β_i^c, $1 \le i \le n$, the sum of the i^{th} row and i^{th} column of matrix B, respectively. As B

is skew symmetric, $\beta_i^r = -\beta_i^c$, $1 \le i \le n$. Then, for every $\pi \in \mathcal{S}_n$, the following equalities holds:

$$Z(A,B,\pi) = \sum_{i=1}^n \sum_{j=1}^n a_{\pi(i)\pi(j)} b_{ij} = \sum_{i=1}^n \alpha_{\pi(i)}^r \beta_i^r + \sum_{i=1}^n \alpha_{\pi(i)}^c \beta_i^c = \sum_{i=1}^n \left(\alpha_{\pi(i)}^r - \alpha_{\pi(i)}^c\right) \beta_i^r$$

Thus, solving QAP(A,B) is equivalent to solving the following minimization problem

$$\min_{\pi \in \mathcal{S}_n} \sum_{i=1}^n \left(\alpha_{\pi(i)}^r - \alpha_{\pi(i)}^c\right) \beta_i^r$$

According to Proposition 1.1, the last minimization problem can be solved by sorting both $n \times 1$ vectors $(\alpha_i^r - \alpha_i^c)$ and (β_i). This takes $O(n \log n)$ time, whereas the vector (β_i^r) can be found in $O(n^2)$ time. Here we assume that the row and column vectors generating matrix A, (α_i^c) and (α_i^r), are given. ■

The next proposition is another early result due to Krushevski [109]. It discusses a polynomially solvable case of QAP where the algebraic structure of the coefficient matrices A and B allows to represent the objective function of the QAP as a quadratic form. For more details on this special case the reader is referred to the original reference in Russian and to Rendl [151].

Proposition 3.12 (Krushevski 1965, [109])
Let real numbers $u_1 \le u_2 \le \cdots \le u_n$, $v_1 \ge v_2 \ge \cdots \ge v_n$, $u = \sum_{i=1}^n u_i$, $v = \sum_{i=1}^n v_i$, and α_1, α_2, β_1, β_2 be given. Let the $n \times n$ matrices $A = (a_{i,j})$ and $B = (b_{ij})$ be defined by $a_{ij} = \beta_1 v_i + \beta_2 v_j + v_i v_j$ and $b_{ij} = \alpha_1 u_i + \alpha_2 u_j + u_i u_j$, respectively. If $\sum_{i=1}^n u_i v_i \ge -\Delta/2$ holds, where

$$\Delta = (\alpha_1 + \alpha_2)v + (\beta_1 + \beta_2)u + n(\alpha_1\beta_1 + \alpha_2\beta_2)\ ,$$

then the identity permutation yields an optimal solution to QAP(A,B).

Proof. For any fixed $\pi \in S_n$ consider $\langle V^\pi, U\rangle$, where $U = (u_i)$ and $V = (v_i)$, and rewrite the objective function of QAP(A,B) as

$$Z(A,B,\pi) = \sum_{i=1}^n \sum_{j=1}^n a_{\pi(i)\pi(j)} b_{ij} = \langle V^\pi, U\rangle^2 + \Delta\langle V^\pi, U\rangle + (\alpha_1\beta_2 + \alpha_2\beta_1)uv. \quad (3.10)$$

The function $f(x) = x^2 + \Delta x + b$ increases for $x \ge -\Delta/2$. Since by assumption, $-\Delta/2 \le \langle V, U\rangle \le \langle V^\pi, U\rangle$ holds, the expression in the right hand side of (3.10) is minimized by $\pi = id$. ■

Remark. Notice that in general Proposition 3.12 cannot be derived as a corollary from our previous results, although the structure of both matrices A, B is quite simple. Namely, $A = A_1 + A_2$ and $B = B_1 + B_2$, where $A_1 = (\alpha_1 v_i + \alpha_2 v_j)$, $A_2 = (v_i v_j)$,

$B_1 = (\beta_1 u_i + \beta_2 u_j)$, $B_2 = (u_i u_j)$, for $1 \le i, j \le n$. Thus, A is sum of two matrices $A_1 \in$ Sum and $A_2 \in$ Product(Sym) and similarly, $B = B_1 + B_2$, where $B_1 \in$ Sum and $B_2 \in$ Product(Sym). In this case, one could think of writting the objective function $Z(A, B, \pi)$ as sum of the objective functions of four other QAPs with a "nicer", already investigated, structure. Namely, $Z(A, B, \pi) = Z(A_1, B_1, \pi) + Z(A_1, B_2, \pi) + Z(A_2, B_1, \pi) + Z(A_2, B_2, \pi)$, for all $\pi \in \mathcal{S}_n$. We can minimize each of $Z(A_i, B_j, \pi)$, $i, j = 1, 2$, in the above equation, by applying point (b) of Theorem 3.11. Then, we would be lucky to have also minimized $Z(A, B, \pi)$, only if problems $QAP(A_i, B_j)$, $i, j = 1, 2$, have a common optimal solution. But, as described in the proof of Theorem 3.11, this depends on the monotonicity of matrices A_i and B_i, $i = 1, 2$, i.e., on the algebraic signs of coefficients α_i and β_i, $i = 1, 2$, and u_i, v_i, $1 \le i \le n$. In the case that α_i, β_i, $i = 1, 2$, and u_i, v_i, $i = 1, 2, \ldots, n$, have all the same sign, the problem is obviously solved by applying Theorem 3.8, even when condition $\sum_{i=1}^{n} u_i v_i \ge -\Delta/2$ is dropped. Let us formulate this as a corollary:

Corollary 3.13 *Let $u_1 \le u_2 \le \cdots \le u_n$, $v_1 \ge v_2 \ge \cdots \ge v_n$, $u = \sum_{i=1}^{n} u_i$, $v = \sum_{i=1}^{n} v_i$, and α_1, α_2, β_1, β_2 be real numbers. Let the $n \times n$ matrices $A = (a_{ij})$ and $B = (b_{ij})$ be defined by $a_{ij} = \beta_1 v_i + \beta_2 v_j + v_i v_j$ and $b_{ij} = \alpha_1 u_i + \alpha_2 u_j + u_i u_j$, respectively. If the numbers α_i, β_i, $i = 1, 2$, and u_i, v_i, $i = 1, 2, \ldots, n$, have all the same sign, then the identity permutation yields an optimal solution to QAP(A,B).*

Proof. Notice that, under the above conditions, one of the matrices A, B is left-higher graded, whereas the other one is right-lower graded. ■

3.4 QAPs on Monge-like matrices

In this section, the computational complexity of QAPs is investigated where both coefficient matrices are restricted to Monge matrices or to subclasses of Monge. Our interest in these restricted versions of the QAP is due to the well known fact that many "difficult" combinatorial optimization problems become "easy" to solve when the input is restricted to Monge or Monge-like matrices (or other mathematical structures with Monge or Monge-like properties). The *traveling salesman problem* (TSP) and the *transportation problem* are two classical examples. For a detailed overview on the role of Monge and Monge-like properties in combinatorial optimization the reader is referred to [32]. It will turn out that in most of the cases QAP(A,B) restricted to Monge matrices remains $\mathcal{NP}$-hard. Anyway, if the coefficient matrices A and B fulfill also additional conditions, polynomially solvable cases arise.

Most of the results in this section are related to QAPs whose coefficient matrices belong to one of the classes Monge, NegMonge=Anti-Monge, Product(Sym), NegProduct(Sym), Chess and NegChess. First, notice that the following inclusion holds:

$$\text{Chess} \subseteq_\pi \text{Product(Sym)} \subseteq_\pi \text{NegMonge}\,. \tag{3.11}$$

	Monge	NegProduct(Sym)	NegChess
Monge	NP	NP	NP
NegProduct(Sym)	NP	NP	NP
NegChess	NP	NP$^{(d)}$	poly$^{(a)}$
Anti-Monge	???	???	poly$^{(b)}$[2]
Product(Sym)	???	poly$^{(c)}$	poly
Chess	poly	poly	poly

Table 3.1: The computational complexity of QAP on Monge-like matrices.

Equivalently, we have NegChess$\subseteq_\pi$NegProduct(Sym)$\subseteq_\pi$Monge.
The results on such QAPs are summarized in Table 3.1. An entry "poly" means that the QAP with input matrices taken from the corresponding row and column is solvable in polynomial time, an entry "NP" means that the corresponding problem is $\mathcal{NP}$-hard and an entry "???" means that the computational complexity of the corresponding problem is unknown. When reading the table, note that two Negs cancel each other and that the table indeed contains all information. E.g., the complexity of QAP with both input matrices in Product(Sym) is found at the intersection of the NegProduct(Sym)-row with the NegProduct(Sym)-column. Because of the inclusion relations (3.11), Table 3.1 has much additional structure. An "NP" entry in the upper half makes all other entries in the upper half that lie above or to the left of it to "NP"-entries. "poly"-entries produce other "poly"-entries below and to the right of them. An analogous statement holds for the lower half. Moreover, notice that both the upper and the lower half of Table 3.1 are symmetric with respect to the corresponding main diagonals. This is immediately seen for the upper half of the table. The symmetry of the lower half of the table is due to the equivalence of problems QAP(-A,B) and QAP(A,-B), for arbitrary matrices A and B. Due to this structure, the four results marked by superscripts would imply all the others in Table 3.1. The results marked by the superscripts (a), (b), (c) and (d) are proved in Theorems 3.14, 3.15, 3.16 and 3.17, respectively.

Theorem 3.14 *The problems* Chess×Chess *and* Chess×NegChess *are solvable in polynomial time.*

Proof. The objective function $Z(A,B,\pi)$ of the problem Chess×Chess can be written as

$$Z(A,B,\pi)=\sum_{i=1}^{n}\sum_{j=1}^{n}(-1)^{\pi(i)+\pi(j)}(-1)^{i+j}=\left(\sum_{i=1}^{n}(-1)^{i+\pi(i)}\right)^2=(n-4m_\pi)^2\ ,$$

[2]The problem Chess(Even)×Monge(Even) is proven to be polynomially solvable, whereas the complexity of Chess(Odd)×Monge(Odd) is still an open question.

where A, B are two $n \times n$ chessboard matrices, $\pi \in \mathcal{S}_n$ and m_π is the cardinality of the set $\{i \in \{1, 2, \ldots, n\} \mid \pi(i) \text{ is odd and } i \text{ is even}\}$. It is easy to check that, for each $1 \le m \le \lfloor n/2 \rfloor$, there exists a permutation $\pi \in \mathcal{S}_n$ such that $m = m_\pi$. Hence, the optimal value of QAP(A,B) equals $\min\{(n-4m)^2 | 0 \le m \le \frac{n}{2}\}$. This optimal value equals 1 for odd n, 0 for n divisible by four and 4 for $n \equiv 2(mod\, 4)$. Denote by m_0 the value of m for which the above minimum is achieved. Then, each permutation $\pi \in \mathcal{S}_n$ with $m_0 = m_\pi$ is an optimal solution to QAP(A,B).

For the second problem Chess×NegChess, the identity permutation is an optimal solution as it yields an objective function value equal to $-n^2$ which is obviously the optimal one, since it is a lower bound for the values of the objective function. ■

Next, let us consider the problem Monge×Chess. Notice that this problem is a restricted version of Monge×Product(Sym) (referred as an open problem in Table 3.1). We show that problem Monge×Chess of even size is a constant permutation QAP and give the constant permutation. It is worthy to note that this result was implicitly proved by Pferschy, Rudolf and Woeginger [143]. The authors investigate the *maximum balanced bisection* problem for Monge matrices and show that it is solvable in polynomial time, if the size of the problem is even. An optimal bisection does not depend on the instance of the Monge matrix; it always consists of the first half of the rows (respectively columns) versus the second half of the rows (respectively columns). This bisection problem is equivalent to the problem Monge×Chess. The proof given below illustrates the use of the cone of the nonnegative Monge matrices for dealing with constant permutation QAPs.

Theorem 3.15 *Consider QAP(A,B) where A and B are two $n \times n$ matrices, $A \in$ Monge and $B \in$ Chess. If n is even, $n = 2m$, the permutation π_* defined below is an optimal solution to QAP(A,B):*

$$\pi_*(x) = \begin{cases} i & \text{if } \; x = 2i-1\ , \;\; 1 \le i \le m \\ m+i & \text{if } \;\; x = 2i\ , \;\; 1 \le i \le m \end{cases}$$

Proof. First, observe that w.l.o.g. we can assume matrix A to have nonnegative entries. Otherwise, we can add a large enough constant to all entries of A without changing its combinatorial structure. Moreover, the resulting QAP would have the same optimal solutions as the given one. Next, notice that QAP(A,B) is equivalent to QAP(A,B$^{(1)}$), where $B^{(1)} = \frac{1}{2}(B + I)$ and I is the $n \times n$ matrix of all ones. (These problems even have a common set of optimal solutions.) This is easily seen, considering Observation 3.1 and the fact that, for a fixed matrix A and $\pi \in \mathcal{S}_n$, $Z(A, I, \pi)$ is a constant independent on π. So, we prove the theorem for QAP(A,B$^{(1)}$) instead of proving it for QAP(A,B), where $B^{(1)}_{ij} = \left(b^{(1)}_{ij}\right)$ and

$$b^{(1)}_{ij} = \begin{cases} 1 & \text{if } \;\; i+j \;\; \text{is even} \\ 0 & \qquad \text{otherwise} \end{cases}$$

We show that permutation π_* is an optimal solution of QAP(D,B$^{(1)}$), for all 0-1 matrices D which generate some extremal ray of the cone of the nonnegative $n \times n$ Monge matrices as defined in Section 3.2. Then, by exploiting Proposition 3.2 and Observation 3.1, this optimality result carries over to all Monge matrices.

There are four QAP instances to be investigated, corresponding to the four different types of 0-1 matrices generating the cone of nonnegative Monge matrices. In each case, we show that the objective function value corresponding to permutation π_* is the optimal one.

Case 1: $D = H^{(p)}$, $1 \le p \le n$.
The following equalities shows that problem QAP($H^{(p)}, B^{(1)}$) is a constant QAP:

$$Z(H^{(p)}, B^{(1)}, \pi) = \sum_{j=1}^{n} h^{(p)}_{p,\pi(j)} b^{(1)}_{i_0,j} = \sum_{j=1}^{n} b^{(1)}_{i_0,j} = |J| = n/2 \tag{3.12}$$

where i_0 is such that $\pi(i_0) = p$ and $J = \{j \mid j \in \{1,2,\ldots,n\}$ and $j + i_0$ is even$\}$. (In the above equality the fact that n is even is essentially used.) Thus, each permutation π is an optimal solution of QAP($H^{(p)}, B^{(1)}$) and so does π_*.

Case 2: $D = V^{(q)}$, $1 \le q \le n$.
Analogously to Case 1, it can be shown that problem QAP($V^{(q)}, B^{(1)}$) is a constant QAP.

Case 3: $D = L^{(p,q)}$, $1 \le p, q \le n$.
In this case the objective function is given by the following simple formula, for all $\pi \in \mathcal{S}_n$:

$$Z(L^{(p,q)}, B^{(1)}, \pi) = \sum_{i=n-p+1}^{n} \sum_{j=1}^{q} b^{(1)}_{\pi^{-1}(i)\pi^{-1}(j)} = |M_1||M_2| + (p - |M_1|)(q - |M_2|) \;, \tag{3.13}$$

where π^{-1} is the inverse permutation to π and M_1, M_2 are sets defined as follows:

$$M_1 = \Big\{ x \in \{n-p+1, n-p+2, \ldots, n\} \;\mid\; \pi^{-1}(x) \text{ is even} \Big\}$$

$$M_2 = \Big\{ x \in \{1, 2, \ldots, q\} \;\mid\; \pi^{-1}(x) \text{ is even} \Big\}$$

Let us apply formula (3.13) to calculate $Z(L^{(p,q)}, B^{(1)}, \pi_*)$. First, notice that for the permutation π_*^{-1} inverse to π_* we have:

$$\pi_*^{-1}(x) = \begin{cases} 2x - 1 & \text{if} \quad 1 \le x \le m \\ 2(x-m) & \text{if} \quad m+1 \le x \le n \end{cases}$$

Define a new matrix $B^{(2)} = \left(b^{(2)}_{ij}\right)$ by $b^{(2)}_{ij} = b^{(1)}_{\pi_*^{-1}(i)\pi_*^{-1}(j)}$, $1 \le i, j \le n$:

$$b^{(2)}_{ij} = \begin{cases} 1 & \text{if} \quad i \le m,\; j \le m \quad \text{or} \quad i > m,\; j > m \\ 0 & \qquad\qquad \text{otherwise} \end{cases}$$

Considering the equalities $Z(L^{(p,q)}, B^{(1)}, \pi_*) = Z(L^{(p,q)}, B^{(2)}, id) = \sum_{i=n-p+1}^{n} \sum_{j=1}^{q} b_{ij}^{(2)}$ and the structure of matrix $B^{(2)}$ we get:

$$Z(L^{(p,q)}, B^{(1)}, \pi_*) = \begin{cases} 0 & \text{if } p \le m \;\; q \le m \\ (p-m)q & \text{if } p > m \;\; q \le m \\ (q-m)p & \text{if } p \le m \;\; q > m \\ (p-m)m + (q-m)m & \text{if } p > m \;\; q > m \end{cases} \tag{3.14}$$

Now, let us show that $Z(L^{(p,q)}, B^{(1)}, \pi) \ge Z(L^{(p,q)}, B^{(1)}, \pi_*)$, for all $\pi \in \mathcal{S}_n$. This can be done by distinguishing the four following subcases:

(a) $p \le m$, $q \le m$. As the entries of matrices $L^{(p,q)}$ and $B^{(1)}$ are nonnegative, $Z(L^{(p,q)}, B^{(1)}, \pi) \ge 0 = Z(L^{(p,q)}, B^{(1)}, \pi_*)$, for all $\pi \in \mathcal{S}_n$.

(b) $p > m$, $q \le m$. Checking that the following inequalities hold, for all $\pi \in \mathcal{S}_n$, completes the proof in this case:

$$Z(L^{(p,q)}, B^{(1)}, \pi) = |M_1||M_2| + (p - |M_1|)(q - |M_2|) = pq - |M_1|(q - |M_2|) -$$

$$|M_2|(p - |M_1|) \ge pq - m(q - |M_2|) - m|M_2| = q(p-m) = Z(L^{(p,q)}, B^{(1)}, \pi_*)$$

(c) $p \le m$, $q > m$. Analogously as in (b) it can be shown that $Z(L^{(p,q)}, B^{(1)}, \pi) \ge p(q-m)$, for all $\pi \in \mathcal{S}_n$.

(d) $p > m$, $q > m$. We show that $Z(L^{(p,q)}, B^{(1)}, \pi) \ge (p-m)m + (q-m)m$, for all $\pi \in \mathcal{S}_n$. First notice that $p - m \le |M_1| \le m$ and $q - m \le |M_2| \le m$. Now, it is elementary to see that the function f defined as

$$f\colon \{p-m, p-m+1, \ldots, m\} \times \{q-m, q-m+1, \ldots, m\} \to \mathbb{R}^+$$

$$(x, y) \mapsto xy + (p-x)(q-y)\ ,$$

achieves its minimum either at point $(m, q-m)$ or at point $(p-m, m)$. The minimum value is $f(m, q-m) = f(p-m, m) = (p-m)m + (q-m)m$. Considering equality (3.13) we have $Z(L^{(p,q)}, B^{(1)}, \pi) = f(|M_1|, |M_2|) \ge (p-m)m + (q-m)m = Z(L^{(p,q)}, B^{(1)}, \pi_*)$.

Case 4: $D = R^{(p,q)}$, $1 \le p, q \le n$.
In this case the proof is completely analogous to the proof in Case 3. ■

The complexity of Monge×Chess remains open for odd n. The proof of Theorem 3.15 essentially exploits the fact that size of the problem is even. Also the exchange argument in [143], used for solving the balanced bisection problem to optimality, fails for problems of odd size. Moreover, notice that the proof of Theorem 3.15 strongly relies on the fact that the considered problem is a constant permutation QAP. Problem Monge×Chess of odd size is not a constant permutation QAP. The following example shows that even the "simpler" problem QAP$(L^{(p,q)}, B^{(1)})$ of odd size, with $B^{(1)}$ as in the proof of Theorem 3.15, is not a constant permutation QAP. That is, for different pairs (p, q), instances of QAP$(L^{(p,q)}, B^{(1)})$ do *not necessarily have* a common optimal solution.

Example 3.1 Consider the 3×3 matrices $L^{(1,2)}$, $L^{(2,1)}$, $B^{(1)}$ and permutations $\pi_1, \pi_2, \pi_3, \pi_4 \in \mathcal{S}_3$, $\pi_1 = \langle 1,3,2 \rangle$, $\pi_2 = \langle 2,3,1 \rangle$, $\pi_3 = \langle 2,1,3 \rangle$ and $\pi_4 = \langle 3,1,2 \rangle$. It is easy to check that

$$Z(L^{(1,2)}, B^{(1)}, \pi_1) = Z(L^{(1,2)}, B^{(1)}, \pi_2) = 0 \ ,$$

$$Z(L^{(1,2)}, B^{(1)}, \pi_3) = Z(L^{(1,2)}, B^{(1)}, \pi_4) = 1 \ ,$$

$$Z(L^{(2,1)}, B^{(1)}, \pi_3) = Z(L^{(2,1)}, B^{(1)}, \pi_4) = 0 \ ,$$

$$Z(L^{(2,2)}, B^{(1)}, \pi_1) = Z(L^{(2,1)}, B^{(1)}, \pi_2) = 1 \ .$$

Moreover, it is also easy to see that π_1 and π_2 are the only optimal solutions to QAP$(L^{(1,2)}, B^{(1)})$, whereas π_3 and π_4 are the only optimal solutions to QAP$(L^{(2,1)}, B^{(1)})$. Thus these two QAPs do *not* have any common optimal solution. ■

Next we formulate and prove the result marked by the superscript (c) in Table 3.1.

Theorem 3.16 *The problem* Product(Sym)×NegProduct(Sym) *is solvable in polynomial time.*

Proof. Consider the $n \times n$ matrices A, B, where $A = (\alpha_i \alpha_j) \in$ Product(Sym), $B = (-\beta_i \beta_j) \in$ NegProduct(Sym). The QAP(A,B) is equivalent to the following maximization problem

$$\max_{\pi \in \mathcal{S}_n} \{ (\sum_{i=1}^{n} \alpha_{\pi(i)} \beta_i)^2 \} \ .$$

With the help of Proposition 1.1, the maximum and the minimum of $\sum_{i=1}^{n} \alpha_{\pi(i)} \beta_i$ over all permutations $\pi \in S_n$ is determined. The maximum of the two corresponding squared values is the optimal value of the given QAP and the corresponding permutation is an optimal solution to it. ■

What about Product(Sym)×Product(Sym)? Even though this problem seems to be quite similar to Product(Sym)×NegProduct(Sym) and its coefficient matrices have the same simple structure, it turns out that this problem is $\mathcal{NP}$-hard. Actually, even a "more restricted" version of QAP, namely Chess×Product(Sym), is $\mathcal{NP}$-hard, as shown by the next theorem. The $\mathcal{NP}$-hardness proof is done by a reduction from the following version of the $\mathcal{NP}$-complete EquiPartition problem (cf. Garey and Johnson [68]).

EquiPartition

Input: A set $\{x_1, \ldots, x_{2n}\}$ of positive integers.

Question: Does there exist a subset $I \subset \{1, 2, \ldots, 2n\}$, $|I| = n$, such that $\sum_{i \in I} a_i = \sum_{i \notin I} a_i$ holds?

Theorem 3.17 *The problem* Chess×Product*(*Sym*) is $\mathcal{NP}$-hard.*

Proof. Start with an instance of EquiPartition and define a $2n \times 2n$ symmetric product matrix $B = (\beta_i \beta_j)$ by $\beta_i = x_i$, $1 \leq i \leq 2n$. Matrix $A = (\alpha_i \alpha_j)$ is given by $\alpha_i = (-1)^i$. Now, consider the instance QAP(A,B) of Chess×Product(Sym). For a given $\pi \in \mathcal{S}_{2n}$, denote $I_\pi = \{\pi(i) : i \text{ is even}\}$. Similarly as in the proof of the last theorem, $Z(A, B, \pi)$ can be written as follows:

$$Z(A,B,\pi) = \left(\sum_{i \in I_\pi} x_i - \sum_{i \notin I_\pi} x_i \right)^2 \qquad \text{for all} \;\; \pi \in \mathcal{S}_{2n} \tag{3.15}$$

Thus, QAP(A,B) is equivalent to the problem $\min\limits_{\pi \in \mathcal{S}_{2n}} \{ (\sum_{i \in I_\pi} x_i - \sum_{i \notin I_\pi} x_i)^2 \}$. We show that the answer to the given instance of the EquiPartition problem is "YES" if and only if the optimal value of the above defined instance of QAP(A,B) is 0 and this completes the proof.

(if) Assume there exists an $I \subset \{1, 2, \ldots, 2n\}$, $|I| = n$, such that $\sum_{i \in I} x_i = \sum_{i \notin I} x_i$. Consider a permutation $\pi_0 \in \mathcal{S}_{2n}$ which permutes the numbers $2i$, $1 \leq i \leq n$, into elements of I, i.e., equality $I_{\pi_0} = I$ holds. Then equality (3.15) implies $Z(A, B, \pi_0) = 0$.

(only if) Assume that the optimal value of QAP(A,B) is 0. Thus, there exists a permutation $\pi_0 \in \mathcal{S}_{2n}$ such that $Z(A, B, \pi_0) = 0$. Equality (3.15) implies $(\sum_{i \in I_{\pi_0}} x_i - \sum_{i \notin I_{\pi_0}} x_i)^2 = 0$. As $|I_{\pi_0}| = n$, set I_{π_0} solves the instance of the EquiPartition problem by "YES". ■

Considering that Chess ⊂ Product(Sym), Theorem 3.17 implies that QAPs of the class Product(Sym)×Product(Sym) are $\mathcal{NP}$-hard. In other words, for two vectors (α_i) and (β_i), minimizing $\left(\sum_{i=1}^n \alpha_{\pi(i)} \beta_i\right)^2$ over all permutations $\pi \in \mathcal{S}_n$ is $\mathcal{NP}$-hard, whereas maximizing the squared sum is polynomially solvable, as shown by Theorem 3.16. The complexity of the problems Monge × Product(Sym) and Monge×NegMonge remains an open question. Obviously, proving either the polynomiality for the later problem, or the $\mathcal{NP}$-hardness for the first one, would answer both open questions and complete Table 3.1.

The following result is a straightforward corollary of Theorem 3.17:

Corollary 3.18 Monge(Sym, rlG)×Monge(Sym) *is $\mathcal{NP}$-hard.*

Proof. By Theorem 3.17, problem Chess×Product(Sym) is $\mathcal{NP}$-hard. Moreover, it is easy to check that the following inclusions hold

NegProduct(Sym)$\subseteq_\pi$Monge(Sym, rlG), NegChess$\subseteq_\pi$Monge(Sym) ■

From Theorem 3.17 it follows also that the problem Monge(Sym)×NegChess is $\mathcal{NP}$-hard. However, it is possible to derive better results for restricted versions of Monge(Sym)×NegChess, where the Monge matrix is somehow "specially structured". For example, the problem Large(Sym)$\times$NegChess can be polynomially solved by a dynamic programming approach. Also Large(Sym)$\times$Chess can be solved by a quite similar dynamic programming scheme. Recall here that Large(Sym)$\times$Chess is a special case of Monge(Sym)$\times$Chess. The complexity of odd sized instances of the latter problem is still an open question.

Theorem 3.19 *The problems* Large(Sym)$\times$NegChess, Large(Sym)$\times$Chess *of size n are solvable in $O(n^2)$ time by dynamic programming.*

Proof. We prove the result only for problem Large(Sym)$\times$NegChess. A completely analogous proof can be given for Large(Sym)$\times$Chess.

Consider QAP(A,B) of size n with $A \in$ Large(Sym) and $B \in$ NegChess. First we make two important remarks.

1. W.l.o.g., we can assume the large matrix A to be left-higher graded. Indeed, if its generating vector is not an increasing one, we can reorder it increasingly. This would result in permuting the rows and the columns of matrix A by the same appropriate permutation, say ψ, to get a matrix $A^{\psi} = (a_{\psi(i),\psi(j)})$. Obviously solving QAP(A,B), is equivalent to solving QAP(A$^{\psi}$,B) as shown by equality $Z(A,B,\pi) = Z(A^{\psi}, B, \psi^{-1} \circ \pi)$, for all $\pi \in \mathcal{S}_n$.

2. Analogously as in the proof of Theorem 3.15, instead of solving QAP(A,B) we may equivalently solve QAP(A,B$'$), where $B' = (b'_{ij})$ and

$$b'_{i,j} = \begin{cases} 0 & \text{if} \quad i+j \text{ is even} \\ 1 & \quad\text{otherwise} \end{cases}$$

In the following we construct a dynamical programming scheme for solving QAP(A,B) with $A \in$Large(Sym,lhG) and $B = B'$. Let (α_i) be the generating vector for A, $\alpha_i \le \alpha_{i+1}$, $1 \le i \le n-1$. Considering that $a_{ij} = a_{ji} = \alpha_i$ for $i \ge j$, the objective function of QAP(A,B) can be rewritten as follows:

$$Z(A,B,\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} b_{\pi^{-1}(i)\pi^{-1}(j)} = 2\sum_{i=2}^{n}\left(\alpha_i \sum_{j=1}^{i-1} b_{\pi^{-1}(i)\pi^{-1}(j)}\right) .$$

Considering the last equality and the structure of matrix B, it is easy to see that for an arbitrary permutation $\pi \in \mathcal{S}_n$, the parities of the values $\pi^{-1}(i)$, $1 \le i \le n$, determine the corresponding objective function value $Z(A,B,\pi)$. Thus, if for two arbitrary permutations $\pi_1, \pi_2 \in \mathcal{S}_n$, the equality $I_{\pi_1} = I_{\pi_2}$ holds, then $Z(A,B,\pi_1) = Z(A,B,\pi_2)$, where $I_{\pi} = \{\pi(i) \colon i \text{ is even}\}$, for $\pi \in \mathcal{S}_n$. Hence, for solving QAP(A,B), it is sufficient to specify the corresponding subset I_{π_0} of an optimal solution π_0 rather than π_0 itself. In other words we are looking for an appropriate set $I_0 \subset \{1,2,\ldots,n\}$, $|I_0| = n$. Obviously, once such a set I_0 is found, we can construct a permutation

$\pi_0 \in \mathcal{S}_n$ such that $I_{\pi_0} = I_0$. π_0 would be an optimal solution to QAP(A,B) [3]. We show that the required set I_0 can be constructed by a dynamic programming approach.

For $2 \le k \le n$, $0 \le s \le \min\{k, \lfloor \frac{n}{2} \rfloor\}$, $0 \le k-s \le \min\{k, \lceil \frac{n}{2} \rceil\}$, consider the auxiliary problems $P(k,s)$ defined as follows:

$$\mathbf{P(k,s)}: \qquad \min_{\substack{\pi\in\mathcal{S}_n \\ |I_\pi \cap\{1,2,\ldots,k\}|=s}} \left\{ 2\sum_{i=2}^{k} \left(\alpha_i \cdot \left| \left\{ j\colon 1 \le j \le i-1,\ \pi^{-1}(i)+\pi^{-1}(j) \text{ is odd} \right\} \right| \right) \right\}$$

Denote by Z(k,s) the optimal value of problem P(k,s). It is easily seen that QAP(A,B) is exactly the same problem as P(n,$\lfloor \frac{n}{2} \rfloor$). Thus, Z(n,$\lfloor \frac{n}{2} \rfloor$) is the optimal value of QAP(A,B).

Next we introduce our dynamic programming scheme. For a feasible pair (k,s) with $s < \min\{k, \lfloor n/2 \rfloor\}$ and $k \ge 3$, an optimal solution of problem P(k,s+1) maps either an odd number or an even number to k. It is easily seen that the following equality holds, for all feasible pairs (k,s) as above:

$$Z(k,s+1) = \min\left\{ 2\alpha_k(s+1) + Z(k-1,s+1), 2\alpha_k(k-1-s) + Z(k-1,s) \right\} \quad (3.16)$$

Now we see that $Z(2,0) = Z(2,2) = 0$, $Z(2,1) = 2\alpha_2$ and $Z(k,0) = 0$, for $2 \le k \le \lceil n/2 \rceil$. Moreover we set $Z(k,0) := +\infty$ for all unfeasible pairs (k,s), $1 \le k,s \le n$. Let us assume that we know $Z(k-1,s)$, for $1 \le s \le n$. Considering that $Z(k,0)$ is also known, we can consecutively calculate $Z(k,1), Z(k,2), \ldots$ by equality (3.16). Obviously, this can be done in $O(n)$ steps which require $O(1)$ time each. Thus, starting with $Z(2,s)$ $(k-1=2)$ and applying the above scheme, the computation of $Z(n,\lfloor n/2 \rfloor)$ takes $O(n^2)$ elementary operations. Then, set I_0 can be specified by backtracking the minima in (3.16) and setting $k \in I_0$ if and only if the corresponding minimum is achieved at the second term in the righthand side of (3.16). ■

According to Table 3.1, problem Anti-Monge×Chess in $\mathcal{NP}$-hard, whereas the complexity of Anti-Monge×NegChess of odd size is unknown. The following straightforward corollary of Theorem 3.19 shows that solvable special cases of the above problems arise if the Anti-Monge matrix has some additional “nice structure”.

Corollary 3.20 *The QAPs* Small(Sym)×Chess *and* Small(Sym)×NegChess *are solvable in $O(n^2)$ time by dynamic programming, where n is the size of the problem.*

Proof. Considering Proposition 3.1 it is easy to see that QAP(A,B) is equivalent to QAP(-A,-B). Next, notice that NegSmall(Sym) = Large(Sym). Thus the problems Small(Sym)×Chess and Small(Sym)×NegChess are equivalent to Large(Sym)×NegChess and Large(Sym)×Chess, respectively. ■

[3] Actually we can construct much more than just one permutation π having the property $I_\pi = I_0$. Namely, there are $\binom{n}{\lfloor n/2 \rfloor} \lfloor n/2 \rfloor!$ such permutations. Clearly, all of them are optimal solutions to QAP(A,B).

3.5 QAPs with circulant and small bandwidth matrices

This section and the next one deal with QAPs restricted to diagonally structured matrices. There are two main reasons which motivate the investigation of QAPs on diagonally structured matrices.
First, several "difficult" combinatorial optimization problems, similar to QAPs with respect to their combinatorial structure, become "easy" when restricted to diagonally structured matrices. As typical examples consider the *shortest Hamiltonian path problem* on circulant matrices and the *traveling salesman problem (TSP)* on matrices with small bandwidth. It is worthy to notice here that the TSP on circulant matrices is still a challenging open problem. (For a detailed discussion on these topics see [116]). The interest on these QAP versions grows while considering the positive results achieved in the 1960's and 1970's on the identification of polynomially solvable cases of QAPs on Toeplitz matrices.
The second reason is related to a large number of problems of both theoretical and practical nature, which can be formulated as QAPs on diagonally structured matrices. As applications with practical relevance let us mention the so called *turbine problem* and different versions of *placement problems.* Some applications will be discussed in detail further on in this chapter. As typical theoretical applications consider the TSP and the *partition into cycles of fixed length.* As such problems are $\mathcal{NP}$-hard in their general formulation, the identification of polynomially solvable restricted versions of them becomes an interesting research direction.

In the current section we consider QAPs one of whose coefficient matrices is either a circulant or has small bandwidth. As we will see, these restrictions on the coefficient matrices are too weak to guarantee the polynomiality of the corresponding versions of QAP. Therefore, we assume the other coefficient matrix to have Monge or Monge-like properties. It will turn out that under these conditions some polynomially solvable cases arise. In the first part of this section the $\mathcal{NP}$-hardness of a QAP with a circulant matrix is derived and a polynomially solvable case of a QAP with bandwidth-k matrices is identified. The latter can also be interpreted as a graph theoretical result. In the second part of this section polynomially solvable cases of the so called *taxonomy problem* are presented.

3.5.1 Two cases of QAPs with circulant and bandwidth-k matrices

The first result in this subsection is a straightforward corollary of Theorem 3.17 showing that CIRCULANT×MONGE is a "difficult" problem.

Corollary 3.21 *The problem* CIRCULANT×MONGE *is $\mathcal{NP}$-hard.*

Proof. Recall that in the proof of Theorem 3.17 we have proven the $\mathcal{NP}$-hardness of problem Product(Sym)$\times$Chess(Even). Clearly, this problem is equivalent to NegProduct(Sym)$\times$NegChess(Even), thus the latter is $\mathcal{NP}$-hard too. Considering the following inclusions completes the proof:

$$\text{NegProduct(Sym)}\subseteq_\pi\text{Monge}\ ,\quad \text{NegChess(Even)}\subseteq_\pi\text{Circulant} \qquad \blacksquare$$

We conjecture that the QAP with circulant matrices, Circulant$\times$Circulant, is $\mathcal{NP}$-hard. Notice that this problem contains the TSP on circulant matrices as a special case. However, in the case that both circulant matrices A and B of QAP(A,B) have a very special structure, polynomially solvable cases arise. The following proposition is a straightforward corollary of a result on the Hamiltonicity of the so called *circulant digraphs with two stripes*. For more details in this topic the reader is referred to [185].

Proposition 3.22 (Yang, Burkard, Çela and Woeginger 1995, [185]*)*
Consider a QAP(A,B) with circulant $n\times n$ matrices A and B. Assume that two of the numbers c_i, $i = 1, 2, \ldots, n-1$ in the definition of the circulant matrix A are zero and the others are strictly positive. Additionally, assume that $b_{ij} > 0$ if $i - j \equiv 1 \pmod n$ and $b_{ij} = 0$ otherwise. Then, it can be decided in $O(\log^4 n)$ whether the optimal value of QAP(A,B) equals zero. If this is the case, an optimal solution can be founded in $O(n)$ time. $\blacksquare$

The next theorem represents a result on the class Monge(Sym)$\times$Bandwidth-k. The considered problem arises as QAP formulation of the *partition into k-cycles problem*, formulated below. Let a weighted graph $G = (V, E)$ be given with vertex set V and edge set E. Let $w: E \to \mathbb{R}$ be a weight function which joins a weight w_{ij} to each edge $(i, j) \in E$. W.l.o.g., we assume that G is complete by assigning a weight equal to ∞ to all non-existent edges. Moreover, let $|V| = k \cdot m$, $m > 1$, $k > 1$. Consider m subgraphs of graph G, $G_i = (V_i, E_i)$, induced by subsets V_i of V respectively, $1 \le i \le m$. Assume that for all $1 \le i \le m$, G_i is a cycle of length k. Moreover, assume that $\cup_{i=1}^k V_i = V$ and $V_i \cap V_j = \emptyset$, $i \ne j$. Such a set of subgraphs $\{G_i: 1 \le i \le m\}$ is called *partition into k-cycles* in G and is denoted by $\mathcal{C}_k$. Its weight is given by

$$w(\mathcal{C}_k) = \sum_{i=1}^{m} \sum_{e\in E_i} w(e)$$

For a given weighted graph $G = (V, E)$, with $|V| = mk$, $k > 1$, $m > 1$, the *partition into k-cycles problem*, shortly *PkCP*, is the problem of finding a partition into k-cycles in G with minimum weight.
This problem can easily be formulated as a QAP. Indeed, denote by A the weighted adjacency matrix of graph G, $A = (w_{ij})$. Let $B^{(k,m)}$ be the adjacency matrix of a

$$B^{(4,3)} = \begin{pmatrix}
\mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0}
\end{pmatrix}$$

Figure 3.1: The matrix B with $k = 4$ and $m = 3$

graph G' on V which consists of m vertex disjoint cycles of length k each, $G'_i = (V'_i, E'_i)$, $1 \le i \le m$, where

$$V_i = \{v_{(i-1)k+j} : j = 1, \ldots, k\} \text{ and}$$

$$E_i = \{(v_{(i-1)k+j}, v_{(i-1)k+j+1}) : j = 1, \ldots, k-1\} \cup \{(v_{(ik}, v_{(i-1)k+1})\}$$

Obviously, $B^{(k,m)}$ is a Bandwidth-k matrix. This definition of the $km \times km$ matrix $B^{(k,m)}$ is valid through the rest of this subsection. As an example consider $k = 4$, $m = 3$. The matrix $B^{(4,3)}$ is presented in Figure 3.1.
Clearly, with these definitions and notations PkCP is equivalent to $QAP(A, B^{(k,m)})$. We show that if A is a symmetric $n \times n$ Monge matrix, $QAP(A, B^{(k,m)})$ is a constant permutation QAP, where $n = km$, k, m are integers and $k > 1$, $m > 1$. Let us denote its constant permutation by $\rho^{(k,m)}$. Thus, $\rho^{(k,m)}$ is an optimal solution to all instances of Monge(Sym)$\times B^{(k,m)}$ with fixed k and m. Before introducing the permutations $\rho^{(k,m)} \in \mathcal{S}_{mk}$, $k > 1$, $m > 1$, let us introduce another permutation π^* which plays a special role through the rest of this chapter.

Definition 3.5 *For $n \in \mathbb{N}$, the permutation $\pi^* \in S_n$ is defined by $\pi^*(i) = 2i - 1$ for $1 \le i \le \lceil \frac{n}{2} \rceil$, and $\pi^*(n+1-i) = 2i$ for $1 \le i \le \lfloor \frac{n}{2} \rfloor$.*
For fixed $k > 1$, $m > 1$, permutation $\rho^{(k,m)} \in \mathcal{S}_{km}$ is given as follows:

$$\rho^{(k,m)}((i-1)k + j) = (i-1)k + \pi^*(j)\ , \quad 1 \le i \le m, \ \ 1 \le j \le k \ \textit{,where } \pi^* \in \mathcal{S}_k .$$

As an example consider $\rho^{(4,3)} = \langle 1, 3, 4, 2, 5, 7, 8, 6, 9, 11, 12, 10 \rangle$.
For the sake of readability let us verbally describe the sequential representation of permutation $\rho^{(k,m)}$. The numbers $1, 2, \ldots, km$ are divided into m groups, which are then consecutively ordered. The first group contains the numbers $1, 2, \ldots, k$, the

next one contains the numbers $k+1, k+2, \ldots, 2k$ and so on, ending up with the last group which contains the numbers $(m-1)k+1, \ldots, km$. The elements of each group are sorted following the same rule defined by permutation π^*: first sort increasingly the elements placed in odd positions within the group and then sort decreasingly the remaining elements. In the above example the groups are $1,2,3,4,\quad 5,6,7,8,$ $9,10,11,12$ and $13,14,15,16$.

Theorem 3.23 *The permutation $\rho^{(k,m)}$ is an optimal solution to $QAP(A, B^{(k,m)})$ of size $n = km$, where matrix $A \in$ MONGE(SYM) and $k > 1$, $m > 1$ are integers.*

The proof is carried through by four lemmas. First, it is shown that the theorem holds for $m = 2$. Then this result is generalized for arbitrary m. The proof of the theorem in the case $m = 2$ is itself reduced into two simpler cases. Namely, it is proven that $\rho^{(k,2)}$ is an optimal solution to $QAP(A, B^{(k,2)})$ in the case that matrix A generates some extremal ray of the cone of nonnegative symmetric Monge matrices of size $2k$. Then, by applying Observation 3.1, this results carries over to all matrices in this cone. We formulate and prove these lemmas below.

Lemma 3.24 *Consider the problem $QAP(S^{(l)}, B^{(k,2)})$ of size $2k$, $k > 1$, where matrix $S^{(l)}$, $1 \le l \le 2k$, generates some extremal ray of the cone of the nonnegative symmetric Monge matrices and is defined in Section 3.2. Permutation $\rho^{(k,2)}$ is an optimal solution to this problem.*

Proof. Consider an instance $QAP(S^{(l)}, B^{(k,2)})$ defined by some fixed integers k, l, where $k > 1$ and $1 \le l \le 2k$. We show that this problem is a constant QAP. That is, the objective function value $Z(S^{(l)}, B^{(k,2)}, \pi)$ is independent on the permutation π. Indeed, for each $\pi \in \mathcal{S}_{2k}$ the objective function of this problem can be written as follows:

$$Z(S^{(l)}, B^{(k,2)}, \pi) = \sum_{j=1}^{k-1} s^{(l)}_{\pi(j)\pi(j+1)} + \sum_{j=k+1}^{2k-1} s^{(l)}_{\pi(j)\pi(j+1)} + s^{(l)}_{\pi(k)\pi(1)} + s^{(l)}_{\pi(2k)\pi(k+1)}\,, \quad (3.17)$$

where $S^{(l)} = \left(s^{(l)}_{ij}\right)$. Denote by i_0 the index in $\{1, 2, \ldots, 2k\}$ such the $\pi(i_0) = l$. Clearly, there are only two terms in (3.17) with row index or column index equal to l. (Obviously, there is no term in (3.17) whose both row and column indices equal l.) Thus, $Z(S^{(i)}, B^{(k,2)}, \pi) = 2$. ■

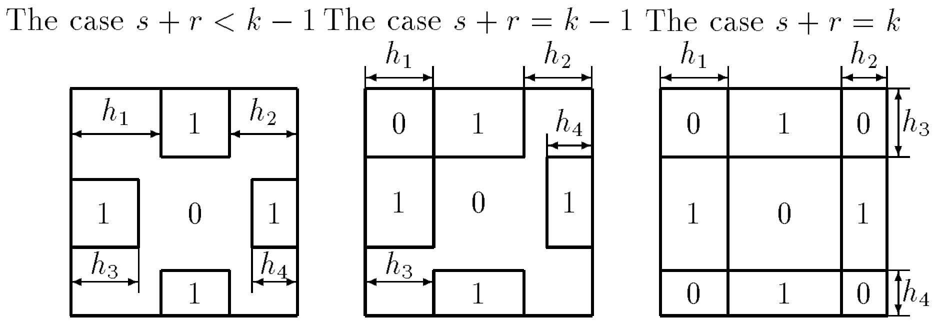
Lemma 3.25 *Consider the problem $QAP(T^{(p,q)}, B^{(k,2)})$ of size $2k$, $k > 1$, where $T^{(p,q)}$, $1 \le p, q \le 2k$, $p + q \le 2k$, is defined in Section 3.2 and generates an extremal ray of the cone of nonnegative symmetric Monge matrices. Permutation $\rho^{(k,2)}$ is an optimal solution to this problem.*

Proof. Consider an instance $QAP(T^{(p,q)}, B^{(k,2)})$ defined by some fixed integers k, p, q, where $k > 1$ and $1 \le p, q \le n$, $p + q \le n$. For the sake of convenience we distinguish two cases whose proofs are completely analogous: $p \le q$ and $p > q$.

Case 1. $p \le q$
First, if $q \le k$, then obviously $Z(T^{(p,q)}, B^{(k,2)}, \rho^{(k,2)}) = 0$. As $Z(T^{(p,q)}, B^{(k,2)}, \pi) \ge 0$ for any $\pi \in \mathcal{S}_{2k}$, $\rho^{(k,2)}$ is an optimal solution of the problem at hand. Thus, w.l.o.g., we assume that $q > k$. Clearly, $p + q \le 2k$ implies then $p < k$. That is, the nonzero entries of $T^{(p,q)}$ have either the row index or the column index larger than k, but not both. Notice moreover, that the permutation $\rho^{(k,2)}$ permutes the elements of the set $\{k+1, \ldots, 2k\}$ to elements of the same set $\{k+1, \ldots, 2k\}$. Therefore, the above placement of the nonzero entries $T^{(p,q)}$ is maintained after permuting it by $\rho^{(k,2)}$. The nonzero entries of matrix $B^{(k,2)}$ may either have both row and column indices larger than k or both row and column indices smaller than or equal to k. Under these conditions only products of entries with row and column indices smaller than or equal to k may contribute with positive values to the objective function value $Z(T^{(p,q)}, B^{(k,2)}, \rho^{(k,2)})$. Moreover, for all $x \in \{1, 2, \ldots, k\}$ we have $\rho^{(k,2)}(k + x) = k + \pi^*(x)$, where $\pi^* \in \mathcal{S}_k$. Therefore, the following equality holds:

$$Z(T^{(p,q)}, B^{(k,2)}, \rho^{(k,2)}) = Z(T^{(p,q-k)}, B^{(k,1)}, \pi^*) \, , \qquad (3.18)$$

where $B^{(k,1)}$ is the $k \times k$ matrix generated by matrix $B^{(k,2)}$ when deleting in it the first k rows and first k columns. Paying a bit of attention to the structure of matrix $B^{(k,1)}$ one realizes that $QAP(T^{(p,q-k)}, B^{(k,1)})$ is nothing else but the traveling salesman problem (TSP) on the symmetric $k \times k$ Monge matrix $T^{(p,q-k)}$. ($T^{(p,q-k)}$ is even an extremal ray of the corresponding cone of nonnegative symmetric Monge matrices.) Supnick [172] has proven that π^* is an optimal solution of the TSP on symmetric Monge matrices. Thus, consequently, π^* is an optimal solution to $QAP(T^{(p,q-k)}, B^{(k,1)})$. For the 0-1 matrices generating the extremal rays of the cone of symmetric nonnegative Monge matrices we can do more than specifying an optimal solution of the corresponding TSP. Namely, we can explicitly give the optimal value of the TSP on this matrices. We compute here below $Z(T^{(s,r)}, B^{(k,1)}, \pi^*)$, for $s + r \le k$, $s, r \ge 1$. (Clearly, in the case that $s = 0$ or $r = 0$ the problem becomes trivial.) For simplicity let us assume that $s \le r$. Everything works analogously in the symmetric case $s > r$. Denote by $T_*^{(s,r)}$ the matrix obtained by $T^{(s,r)}$ when permuting its rows and columns by permutation π^*. Thus, $T_*^{(s,r)} = \left(t^{(s,r)}_{\pi^*(i)\pi^*(j)}\right)$, where $T^{(s,r)} = \left(t^{(s,r)}_{ij}\right)$. It is a matter of elementary calculations to see that matrix $T_*^{(s,r)}$ looks as in Figure 3.2. The 0 (1) entry in the interior of a polygone means that all entries of $T_*^{(s,r)}$ falling inside this polygone are equal to 0 (1). Considering Figure 3.2, it is easy to see that $Z(T_*^{(s,r)}, B^{(k,1)}, id)$ equals $0, 1, 2$ in the case that $s + r$ is smaller than $k - 1$, is equal to $k - 1$ or is equal to k, respectively. Now, considering $Z(T^{(s,r)}, B^{(k,1)}, \pi^*) = Z(T_*^{(s,r)}, B^{(k,1)}, id)$ while setting for $s := p$ and $r := q - k$,



$h_1 = \lceil \frac{k-r}{2} \rceil$, $h_2 = \lfloor \frac{k-r}{2} \rfloor$, $h_3 = \lceil \frac{s}{2} \rceil$, $h_4 = \lfloor \frac{s}{2} \rfloor$

Figure 3.2: The $k \times k$ matrix $T_*^{(s,r)}$ for $s < r$, $s + r \le k$.

equation (3.18) implies the following equality:

$$Z(T^{(p,q)}, B^{(k,2)}, \rho^{(k,2)}) = \begin{cases} 0 & \text{if} \quad p+q < 2k-1 \\ 1 & \text{if} \quad p+q = 2k-1 \\ 2 & \text{if} \quad p+q = 2k \end{cases} \tag{3.19}$$

Consequently, for matrices $T^{(p,q)}$ with $p + q < 2k - 1$, $\rho^{(k,2)}$ is the optimal solution to $QAP(T^{(p,q)}, B^{(k,2)})$. It remains to prove this in the case that $2k - 1 \le p + q \le 2k$. We show that $Z(T^{(p,q)}, B^{(k,2)}, \pi) \ge Z(T^{(p,q)}, B^{(k,2)}, \rho^{(k,2)})$, for any $\pi \in \mathcal{S}_{2k}$, and this completes the proof in the two remaining cases. Consider an arbitrary $\pi \in \mathcal{S}_{2k}$. Let $z_1 \le z_2 \le \ldots \le z_k$ such that $\{\pi(z_i): 1 \le i \le k\} = \{1, 2, \ldots, k\}$ and $y_1 \le y_2 \le \ldots \le y_k$ such that $\{\pi(y_i): k+1 \le i \le 2k\} = \{k+1, k+2, \ldots, 2k\}$. Denote by $T^{(z)}$ and $T^{(y)}$ the matrix resulting from deleting in $T^{(p,q)}$ the rows and columns with indices y_i, $1 \le i \le k$ and z_i, $1 \le z \le k$, respectively. Clearly, $T^{(z)}$ and $T^{(y)}$ are symmetric Monge matrices [4]. Due to the structure of matrix $B^{(k,2)}$ the following equality holds:

$$Z(T^{(p,q)}, B^{(k,2)}, \pi) = Z(T^{(z)}, B^{(k,1)}, \pi_z) + Z(T^{(y)}, B^{(k,1)}, \pi_y) ,$$

where π_z and π_y are defined by the equalities $\pi_z(i) = \pi(z_i)$ and $\pi_y(i) = \pi(y_i) - k$, for $1 \le i \le k$. From the construction, the matrices $T^{(z)}$ and $T^{(y)}$ are of the form $T^{(p_z,q_z)}$ and $T^{(p_y,q_y)}$, respectively, where $p_z + p_y = p$ and $q_z + q_y = q$ and $p_z + q_z \le k$, $p_y + q_y \le k$. Now, the inequality $p+q \ge 2k-1$ implies either $p_z + q_z = k$, $p_y + q_y \ge k-1$ or $p_z + q_z \ge k-1$, $p_y + q_y = k$. Then, similarly as above, we can apply the result of Supnick for the TSP on symmetric Monge matrices. Moreover, in the case that the four coefficients p_z,p_y,q_z,q_y are strictly positive we can apply equality (3.19) and

[4]It is well known and easily seen that submatrices obtained from a symmetric Monge matrix by deleting rows and columns with the same set of indices are again symmetric Monge matrices. See, for example, [32].

obtain:

$$Z(T^{(p,q)}, B^{(k,2)}, \pi) = Z(T^{(p_z,q_z)}, B^{(k,1)}, \pi_z) + Z(T^{(p_y,q_y)}, B^{(k,1)}, \pi_y) \geq$$

$$Z(T^{(p_z,q_z)}, B^{(k,1)}, \pi^*) + Z(T^{(p_y,q_y)}, B^{(k,1)}, \pi^*) = 3 \geq Z(T^{(p,q)}, B^{(k,2)}, \ \rho^{(k,2)})$$

Let us now consider the almost trivial case where one of the four coefficients p_z,p_y,q_z, q_y equals 0. Assume w.l.o.g. that $p_z = 0$. If $p+q = 2k-1$, we have $p_y = p$ and either $p_y + q_y = k$, $q_z = k-1$, or $p_y + q_y = k-1$, $q_z = k$. Thus:

$$Z(T^{(p,q)}, B^{(k,2)}, \pi) = Z(T^{(p_y,q_y)}, B^{(k,1)}, \pi_y) \geq Z(T^{(p_y,q_y)}, B^{(k,1)}, \pi^*) \geq 1 =$$

$$Z(T^{(p,q)}, B^{(k,2)}, \rho^{(k,2)}) .$$

In the remaining case $p+q = 2k$, we again have $p_y = p$ and $q_z = k$, $p_y + q_y = k$. This yields

$$Z(T^{(p,q)}, B^{(k,2)}, \pi) = Z(T^{(p_y,q_y)}, B^{(k,1)}, \pi_y) \geq Z(T^{(p_y,q_y)}, B^{(k,1)}, \pi^*) = 2 =$$

$$Z(T^{(p,q)}, B^{(k,2)}, \rho^{(k,2)} .$$

Case 2. $p > q$
The proof is completely analogous to the proof of Case 1. ■

Lemma 3.26 *Permutation $\rho^{(k,2)}$ is an optimal solution of $QAP(A, B^{(k,2)})$ of size $2k$, where $A \in$ MONGE(SYM).*

Proof. As already argued several times in this chapter, we may w.l.o.g. assume matrix A to have nonnegative entries. Otherwise, we could add a large enough constant to all of its entries. The combinatorial structure of the resulting matrix would remain the same, whereas the resulting QAP would be equivalent to the given one. Under this assumption, the proof follows immediately from the previous Lemmas 3.24 and 3.25. ■

Lemma 3.27 *Permutation $\rho^{(k,m)}$ is an optimal solution to $QAP(A, B^{(k,m)})$ of size mk, where $A \in$ MONGE(SYM).*

Proof. Assume that matrix A has nonnegative entries. We show that for an arbitrary $\pi \in \mathcal{S}_{km}$ the following inequality holds:

$$Z(A, B^{(k,m)}, \pi) \geq Z(A, B^{(k,m)}, \rho^{(k,m)}) . \tag{3.20}$$

To see this we transform the objective function so as to use then Lemma 3.26. Let $x_1^{(j)} < x_2^{(j)} < \ldots < x_k^{(j)}$ such that $\{\pi(x_1^{(j)}), \pi(x_2^{(j)}), \ldots, \pi(x_k^{(j)})\} = \{(j-1)k+1, (j-1)k+2, \ldots, jk\}$, for $1 \leq j \leq m$. Denote $X^{(j)} = \{x_1^{(j)}, \ldots, x_k^{(j)}\}$, for $1 \leq j \leq m$. Denote by $A^{(j)}$ the $k \times k$ matrix resulting from A when deleting its rows and columns with indices in the complement of $X^{(j)}$. Define additionally the permutations $\pi^{(j)} \in \mathcal{S}_k$ by $\pi^{(j)}(i) = \pi(x_i^{(j)}) - (j-1)k$, $1 \le i \le k$. Clearly, the permutation π is uniquely determined by the permutations $\pi^{(j)}$ and the sets $X^{(j)}$, $1 \le j \le m$. Through the rest of the proof the sets $X^{(j)}$ and the permutations $\pi^{(j)}$ related to a permutation π are called its *partial sets* and its *partial permutations*, respectively. Moreover, the matrices $A^{(j)}$ and the problems $QAP(A^{(j)}, B^{(k,1)})$ are called *partial matrices* and *partial problems* of $QAP(A, B^{(k,m)}$, respectively. Observe that the objective function value of $QAP(A, B^{(k,m)})$ is given as sum of the objective function values of its partial problems obtained by the corresponding partial permutations:

$$Z(A, B^{(k,m)}, \pi) = \sum_{j=1}^{m} Z(A^{(j)}, B^{(k,1)}, \pi^{(j)}) \tag{3.21}$$

Next, we describe a procedure for replacing permutation π by a permutation π' with $Z(A, B^{(k,m)}, \pi) \ge Z(A, B^{(k,m)}, \pi')$. This procedure is called *change*, through the rest of the proof. It will turn out that after applying $O((mk)^2)$ consecutive changes to an arbitrary permutation π, π is transformed to $\rho^{(k,m)}$. As in each change the objective function value decreases, equation (3.20) holds and this completes the proof.

Consider two problems of size k: $QAP(A^{(j)}, B^{(k,1)})$ and $QAP(A^{(l)}, B^{(k,1)})$ with $j < l$. Consider additionally the symmetric $2k \times 2k$ Monge matrix $A^{(j,l)}$ obtained from A when deleting all its rows and columns with indices in the complement of $X^{(j)} \cup X^{(l)}$. Consider also a permutation $\pi^{(j,l)} \in \mathcal{S}_{2k}$ defined by

$$\pi^{(j,l)}(i) = \begin{cases} \pi^{(j)}(i) & \text{if} \quad 1 \le i \le k \\ \pi^{(l)}(i) + k & \text{if} \quad k+1 \le i \le 2k \end{cases}$$

For the problem $QAP(A^{(j,l)}, B^{(k,2)})$ of size $2k$ we have:

$$Z(A^{(j,l)}, B^{(k,2)}, \pi^{(j,l)}) = Z(A^{(j)}, B^{(k,1)}, \pi^{(j)}) + Z(A^{(l)}, B^{(k,1)}, \pi^{(l)})\ .$$

On the other side Lemma 3.26 implies that $\rho^{(k,2)}$ is an optimal solution to $QAP(A^{(j,l)}, B^{(k,2)})$. Consider two new sets $\bar{X}^{(j)}$ and $\bar{X}^{(l)}$. $\bar{X}^{(j)}$ consists of the k smallest elements of $X^{(j)} \cup X^{(l)}$ and $\bar{X}^{(l)} = (X^{(j)} \cup X^{(l)}) \setminus \bar{X}^{(j)}$. Moreover, let $\bar{A}^{(j)}$ and $\bar{A}^{(l)}$ be the two $k \times k$ matrices obtained from A by deleting the rows and the columns with indices in the complement of $\bar{X}^{(j)}$ and $\bar{X}^{(l)}$, respectively. Similar arguments as in the proof of Lemma 3.25 yield: $Z(A^{(j,l)}, B^{(k,2)}, \rho^{k,2)}) = Z(\bar{A}^{(j)}, B^{(k,1)}, \pi^*) + Z(\bar{A}^{(l)}, B^{(k,1)}, \pi^*)$. Thus combining the two last equalities we obtain

$$Z(A^{(j)}, B^{(k,1)}, \pi^{(j)}) + Z(A^{(l)}, B^{(k,1)}, \pi^{(l)}) \ge Z(\bar{A}^{(j)}, B^{(k,1)}, \pi^*) + Z(\bar{A}^{(l)}, B^{(k,1)}, \pi^*) \tag{3.22}$$

Now, we replace the sets $X^{(j)}$ and $X^{(l)}$ by sets $\bar{X}^{(j)}$ and $\bar{X}^{(l)}$ respectively. We also replace both $\pi^{(j)}$ and $\pi^{(l)}$ by π^*, whereas the other partial sets and partial permutations remain unchanged. As noticed above, the new partial sets and the new partial permutations determine a new permutation, say π'. Equations (3.22) and (3.21) imply $Z(A,B,\pi') \le Z(A,B,\pi)$.

We apply to π a series of consecutive changes as described above with $(j,l) = (1,2)$, $(1,3)$, ..., $(1,k)$. Due to the definition of a change, after $k-1$ of these changes all current partial permutations are π^* and the current partial set $X^{(1)}$ equals $\{1,2,\ldots,k\}$. By applying to the current permutation $k-2$ new consecutive changes with $(j,l) = (2,3),(2,4),\ldots,(2,k)$, we transform this permutation to a new one with the same partial permutations π^*, the same partial set $X^{(1)}$ and an eventually new partial set $X^{(2)}$. Partial set $X^{(2)}$ equals now $\{k+1,\ldots,2k\}$. Then we continue this process by applying other consecutive changes with $(j,l) = (3,4),(3,5),\ldots,(3,k),\ldots,(k-1,k)$. After this, all final partial permutations equal π^* and the partial sets are given by

$$X^{(j)} = \{(j-1)k+1,(j-1)k+2,\ldots,jk\}, \qquad 1 \le j \le m$$

It is easy to see that the unique permutation determined by these partial sets and these partial permutations is $\rho^{(k,m)}$. ■

3.5.2 Special cases of the taxonomy problem

The *general taxonomy problem* was introduced by Rubinstein [159]. It belongs to the very large class of problems dealing with optimal groupings of related objects. The problem is very complex in its general formulation. For a detailed description and related topics the reader is referred to [160]. Here we only describe a simple version of the general taxonomy problem which can be formulated as a QAP. We simply call it *taxonomy problem* (TP). All the results presented in this section are related to problem TP which is introduced below.

Consider a group of n objects to be partitioned into a given number of subgroups of a fixed size. Assume that each of these objects is identified with an index from 1 up to n. A communication matrix $A = (a_{ij})$ is given, where the entry a_{ij} represents a communication rate between object i and object j, $1 \le i,j \le n$. Moreover, the communication rates between objects belonging to the same group are scaled by a factor depending on the size of the group. The goal is to find a partition of the given objects into groups which minimizes the overall communication rate between all pairs of objects belonging to the same group. Through the rest of this thesis this problem will be referred to as *taxonomy problem* or, shortly, TP.
Notice that in the case that all subgroups have the same size this problem have been considered by Pferschy, Rudolf and Woeginger [143] under the name *balanced k-cut problem*, where k is the number of subgroups. In [143] it is shown that the special case of the balanced k-cut problem where the communication matrix A is a Monge matrix is solved to optimality by the identity permutation.

The taxonomy problem (TP) is a generalization of the set partitioning problem as defined by Grötschel in [80]. Indeed, consider the given objects as the ground set E in a partitioning problem. Assign a cost c_S to each subset S of E: $c_S = |S| \sum_{i,j \in S} a_{ij}$. The traditional set partitioning problem consists of finding a partition of the ground set E into a number of subsets of E such that the overall cost of the subsets involved by the partition is minimized. In the taxonomy problem it is additionally required that the subsets of the partitioning have prespecified cardinalities.

TP can easily be modeled as a QAP. Indeed, let the set $\{1, 2, \ldots, n\}$ represent the n given objects. Let n_1, n_2,..., n_k be the size of the k subgroups in which the objects have to be partitioned. Clearly, $\sum_{i=1}^{k} n_i = n$. Let $f\colon \mathbb{N} \to \mathbb{R}^+$ be a function on the size of the subgroups modeling the connectivity scaling factor within the groups. Finally, let $A = (a_{ij})$ be the *matrix of connectivities*, i.e., a_{ij} is the numerical measure of connectivity of object i with object j, $1 \le i, j \le n$. In order to formulate TP as a QAP we need one more definition, namely, we define the so called *taxonomy* matrices.

Definition 3.6 *Let a function $f\colon \mathbb{N} \to \mathbb{R}$ and the integers n_i, $1 \le i \le k$, $\sum_{i=1}^{k} n_i = n$, be given. An $n \times n$ matrix A is called a* taxonomy *matrix of type $(f, n_1, n_2, \ldots, n_k)$ if it is a quasidiagonal matrix consisting of k blocks of nonzero elements along the diagonal, where the i-th diagonal block is of size $n_i \times n_i$ and all entries within this block are equal to $f(n_i)$.*

As an example consider a 6×6 taxonomy matrix A of type $(f, 1, 3, 2)$, where f is given by the formula $f(x) = 2x$:

$$A = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 6 & 6 & 6 & 0 & 0 \\ 0 & 6 & 6 & 6 & 0 & 0 \\ 0 & 6 & 6 & 6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 4 & 4 \end{pmatrix}$$

With these notations and definitions, it is clear that the problem TP with the above described input data is mathematically equivalent to QAP(A,B), where A is the matrix of connectivities and B is a taxonomy matrix of type $(f, n_1, n_2, \ldots, n_k)$. The following theorem, formulated by Rubinstein in [161], describes three polynomially solvable cases of this QAP version. Our proof exploits again the fact that the classes of nonnegative matrices in Monge and Monge(rlG) form cones, respectively. Clearly, we can take advantage of this combinatorial structure only because the corresponding solvable QAP versions are constant permutation QAPs.

Theorem 3.28 (Rubinstein 1994, [161])
Let A be a Monge matrix and B be a taxonomy matrix of type $(f, n_1, n_2, \ldots, n_k)$ where f is a nonnegative function. Consider moreover the three following conditions on the problem input:

(a) $n_1 = n_2 = \cdots = n_k$

(b) A is right-lower graded matrix, $n_1 \le n_2 \le \ldots \le n_k$ *and f is a nondecreasing function.*

(c) A is a left-higher graded matrix, $n_1 \le n_2 \le \ldots \le n_k$ *and f is a nondecreasing function.*

In the cases (a) and (b) the identity permutation id is an optimal solution to QAP(A,B). In the case (c) the permutation $\phi = \langle n, n-1, \ldots, 2, 1\rangle$ *is an optimal solution to QAP(A,B).*

Proof. First, notice that analogously as in the proof of Theorem 3.15, we may assume matrix A to have nonnegative entries. Moreover, recall that according to Propositions 3.2, the extremal rays of the cone of the $n \times n$ Monge matrices with nonnegative entries are generated by the $0-1$ matrices $H^{(p)}$, $V^{(q)}$, $1 \le p \le n$ and $L^{(p,q)}$, $R^{(p,q)}$, $1 \le p, q \le n$, introduced in Section 3.2. According to Corollary 3.6, the extremal rays of the cone of the right-lower graded $n \times n$ Monge matrices are generated by the $0-1$ matrices $E^{(p,q)}$, $1 \le p, q \le n$, introduced in Section 3.2.

Hence, in case (a) it is sufficient to prove the theorem only for matrix A coinciding with one of the matrices $H^{(p)}$, $V^{(q)}$, $L^{(p,q)}$ and $R^{(p,q)}$, $1 \le p, q \le n$. Analogously, in case (b) it is sufficient to prove the theorem only for a matrix A coinciding with one of the matrices $E^{(p,q)}$, $1 \le p, q \le n$. Then, applying Observation 3.1, the results carry over to all matrices in the respective cones. We will see that the result in case (c) follows quite easily from the result in case (b).

Proof of (a) Consider a QAP(A,B) of size $n = k \cdot n_1$, where B is a taxonomy matrix of type $(f, n_1, n_2, \ldots, n_k)$, $n_1 = n_2 = \ldots = n_k$ and f is a nonnegative function. Let $s_0 = 0$ and $s_t = \sum_{i=1}^{t} n_i$, for $1 \le t \le k$. Then, the objective function $Z(A, B, \pi)$ can be written as follows:

$$Z(A, B, \pi) = f(n_1) \sum_{t=1}^{k} \Big(\sum_{i=s_{t-1}+1}^{s_t} \sum_{j=s_{t-1}+1}^{s_t} a_{\pi(i)\pi(j)} \Big) \tag{3.23}$$

We consider separately the following four cases:

Case 1. $A = V^{(p)}$, $1 \le p \le n$.
Consider an arbitrary $\pi \in \mathcal{S}_n$ and let s_h be the unique element among $s_0, s_1, \ldots, s_{k-1}$ for which $s_h + 1 \le \pi^{-1}(p) \le s_h$, where π^{-1} is the inverse permutation of π. Then, replacing A by $V^{(p)}$ in (3.23) we get:

$$Z(V^{(p)}, B, \pi) = f(n_1) \sum_{j=s_h+1}^{s_{h+1}} v^{(p)}_{p\pi(j)} = n_1 f(n_1) \,,$$

where $V^{(p)} = \left(v_{ij}^{(p)}\right)$. Thus, $QAP(V^{(p)}, B)$ is a constant QAP and therefore id is an optimal solution to it.

Case 2. $A = H^{(p)}$, $1 \le p \le n$.
Analogously as in Case 1, it can be shown that $QAP(H^{(p)}, B)$ is a constant QAP.

Case 3. $A = R^{(p,q)}$, $1 \le p, q \le n$.
For a given $\pi \in \mathcal{S}_n$, define two vectors $(x_t^{(\pi)})$, $(y_t^{(\pi)})$ as follows:

$$\left.\begin{array}{rcl} x_t^{(\pi)} & = & |\{s_{t-1}+1 \le i \le s_t \ : \ 1 \le \pi(i) \le p\}| \\ y_t^{(\pi)} & = & |\{s_{t-1}+1 \le i \le s_t \ : \ n-q+1 \le \pi(i) \le n\}| \end{array}\right. \quad \text{for} \quad 1 \le t \le k$$

Then, the objective function in (3.23) can be written as

$$Z(R^{(p,q)}, B, \pi) = f(n_1) \sum_{t=1}^{k} x_t^{(\pi)} y_t^{(\pi)} \ .$$

Thus, $QAP(R^{(p,q)}, B)$ is equivalent to the following minimization problem:

$$\min_{\pi \in \mathcal{S}_n} \sum_{t=1}^{k} x_t^{(\pi)} y_t^{(\pi)} \tag{3.24}$$

By definition, the vectors $\left(x_t^{(\pi)}\right)$, $\left(y_t^{(\pi)}\right)$ have nonnegative integer entries and fulfill the following conditions:

(i) $\sum\limits_{t=1}^{k} x_t^{(\pi)} = p, \qquad \sum\limits_{t=1}^{k} y_t^{(\pi)} = q$

(ii) $0 \le x_t^{(\pi)}, y_t^{(\pi)} \le n_t$, $1 \le t \le k$

Relaxing the condition that the vectors $\left(x_t^{(\pi)}\right)$, $\left(y_t^{(\pi)}\right)$ are related to some permutation $\pi \in \mathcal{S}_n$ as described above, we get the following relaxation P1 of the minimization problem (3.24)

$$\text{(P1)} \qquad \begin{array}{ll} \min & \sum_{t=1}^{k} x_t y_t \\ & \\ \text{s.t.} & \sum_{t=1}^{k} x_t = p \\ & \sum_{t=1}^{k} y_t = q \\ & 0 \le x_t, y_t \le n_t, \quad 1 \le t \le k \\ & x_t, y_t \text{ integer}, \quad 1 \le t \le k \end{array}$$

Obviously, the optimal value of problem P1 is a lower bound for the optimal value of $QAP(R^{(p,q)}, B)$. We show that vectors $\left(x_t^{(id)}\right)$ and $\left(y_t^{(id)}\right)$ corresponding to the identity permutation id yield an optimal solution to P1. This would then imply that the identity permutation id is an optimal solution to $QAP(R^{(p,q)}, B)$. Indeed, vectors $\left(x_t^{(id)}\right)$ and $\left(y_t^{(id)}\right)$ are given as follows:

$$x_t^{(id)} = \min\left\{n_t, \max\{p - s_{t-1}, 0\}\right\}, \quad y_t^{(id)} = \min\left\{n_t, \max\{q - n + s_t, 0\}\right\}, \quad 1 \le t \le k \ .$$

It takes a bit of attention to see that these vectors are derived by starting with $t = 1$ and setting the entries $x_t^{(id)}$, $y_{k-t+1}^{(id)}$ as large as possible and then increasing t by 1. This procedure is repeated until $t = k$. As an easy consequence of Proposition 1.1, this construction yields an optimal solution to problem P1. The proof of this easily seen fact, which relies on a simple exchange argument, is omitted here because of its simplicity.

Case 4. $A = L^{(p,q)}$, $1 \le p, q \le n$.
Analogous to the proof in Case 3.

(b) Consider a QAP(A,B) of size $n = \sum_{t=1}^{k} n_t$ where B is a taxonomy matrix of type $(f, n_1, n_2, \ldots, n_k)$, f is a nondecreasing function and $A = E^{(p,q)}$. For any $\pi \in \mathcal{S}_n$, the objective function $Z(A, B, \pi)$ of this QAP(A,B) is given as follows:

$$Z(A,B,\pi) = \sum_{t=1}^{k} \left(f(n_t) \cdot \sum_{i=s_{t-1}+1}^{s_t} \sum_{j=s_{t-1}+1}^{s_t} a_{\pi(i)\pi(j)} \right) \tag{3.25}$$

where s_t, $0 \le t \le k$, are defined as in the proof of (a). Set $p' = n - p$ and $q' = n - q$ and consider additionally the matrix $C^{(p',q')}$, as defined in Section 3.2. Clearly, $E^{(p,q)} + C^{(p',q')} = I$, where I is the $n \times n$ matrix with all entries equal to 1. From Observation 3.1 follows then that minimizing $Z(E^{(p,q)}, B, \pi)$ over all permutations $\pi \in \mathcal{S}_n$ is equivalent to maximizing $Z(C^{(p',q')}, B, \pi)$ over $\pi \in \mathcal{S}_n$. That is, these problems have the same set of optimal solutions. For an arbitrary $\pi \in \mathcal{S}_n$, $Z(C^{(p',q')}, B, \pi)$ can be written as follows:

$$Z(C^{(p',q')}, B, \pi) = \sum_{t=1}^{k} \left(f(n_t) \cdot \sum_{i=s_{t-1}+1}^{s_t} \sum_{j=s_{t-1}+1}^{s_t} c_{\pi(i)\pi(j)}^{(p',q')} \right) = \sum_{t=1}^{k} f(n_t) x_t^{(\pi)} y_t^{(\pi)} ,$$

where the vectors $\left(x_t^{(\pi)}\right)$ and $\left(y_t^{(\pi)}\right)$ are defined as follows:

$$\left.\begin{array}{rcl} x_t^{(\pi)} & = & |\{s_{t-1}+1 \le i \le s_t \ : \ p+1 \le \pi(i) \le n\}| \\ y_t^{(\pi)} & = & |\{s_{t-1}+1 \le i \le s_t \ : \ q+1 \le \pi(i) \le n\}| \end{array}\right. \quad \text{for} \quad 1 \le t \le k$$

Thus, $QAP(E^{(p,q)}, B)$ is equivalent to the following maximization problem

$$\max_{\pi \in \mathcal{S}_n} \sum_{t=1}^{k} f(n_t) x_t^{(\pi)} y_t^{(\pi)} \tag{3.26}$$

By definition, the vectors $\left(x_t^{(\pi)}\right)$, $\left(y_t^{(\pi)}\right)$ have nonnegative integer entries and fulfill the following conditions:

(i) $\sum_{t=1}^{k} x_t^{(\pi)} = p'$, $\quad \sum_{t=1}^{k} y_t^{(\pi)} = q'$

(ii) $0 \le x_t^{(\pi)}, y_t^{(\pi)} \le n_t$, $1 \le t \le k$

Similarly as in the proof in Case 3 of (a), the maximization problem (3.26) can be relaxed to get the following problem P2:

$$(\mathbf{P2}) \qquad \begin{array}{ll} \max & \sum_{t=1}^{k} f(n_t) x_t y_t \\ & \\ s.t. & \sum_{t=1}^{k} x_t = p' \\ & \sum_{t=1}^{k} y_t = q' \\ & 0 \le x_t, y_t \le n_t \quad 1 \le t \le k \\ & x_t, y_t \text{ integer} \quad 1 \le t \le k \end{array}$$

Obviously, the optimal value of problem P2 is an upper bound for the maximum value of $Z(C^{(p',q')}, B, \pi)$ over $\pi \in \mathcal{S}_n$. We show that the vectors $\left(x_t^{(id)}\right)$, $\left(y_t^{(id)}\right)$ corresponding to the identity permutation *id* yield an optimal solution to P2. Clearly, this would imply permutation *id* to be an optimal solution to the maximization problem (3.24) and equivalently to $QAP(E^{(p,q)}, B)$. The vectors $\left(x_t^{(id)}\right)$, $\left(y_t^{(id)}\right)$ are given as follows:

$$x_t^{(id)} = \min\left\{n_t, \max\{0, p' - n + s_t\}\right\}, \quad 1 \le t \le k,$$

$$y_t^{(id)} = \min\left\{n_t, \max\{0, q' - n + s_t\}\right\}, \quad 1 \le t \le k.$$

The fact that f is nondecreasing together with Proposition 1.1 implies that $\left(x_t^{(id)}\right)$, $\left(y_t^{(id)}\right)$ yield an optimal solution to problem P2. Again, the proof of this fact, which relies on a simple exchange argument, is omitted because of its simplicity.

(c) Consider the QAP(A,B) of size n with $A \in$ Monge(lhG), B being a taxonomy matrix of type $(f, n_1, n_2, \ldots, n_k)$ and f being a nondecreasing function. Consider moreover, the permuted matrix $A^\phi = (a_{\phi(i)\phi(j)})$, where $\phi = \langle n, n-1, \ldots, 2, 1\rangle$. The following inequality shows that A^ϕ is a Monge matrix:

$$a_{\phi(i),\phi(j)} + a_{\phi(i+1)\phi(j+1)} = a_{n-i+1\,n-j+1} + a_{n-i\,n-j} \le$$

$$a_{n-i+1\,n-j} + a_{n-i\,n-j+1} = a_{\phi(i+1)\phi(j)} + a_{\phi(i)\phi(j+1)}, \quad 1 \le i, j \le n-1$$

Moreover, A^ϕ is a left-higher graded matrix, as the following inequalities show:

$$a_{\phi(i)\phi(j)} = a_{n-i+1\,n-i+j} \le a_{n-i\,n-j+1} = a_{\phi(i+1)\phi(j)}, \quad 1 \le i \le n-1, \ 1 \le j \le n$$

$$a_{\phi(i)\phi(j)} = a_{n-i+1\,n-i+j} \le a_{n-i+1\,n-j} = a_{\phi(i)\phi(j+1)}, \quad 1 \le i \le n, \ 1 \le j \le n-1$$

Thus, $A^\phi \in$ Monge(lhG) and therefore, permutation *id* is an optimal solution to $QAP(A^\phi, B)$ as proven in (b). Now, the claim follows when considering that the following equality holds:

$$Z(A, B, \phi \circ \pi) = Z(A^\phi, B, \pi), \quad \pi \in \mathcal{S}_n \qquad \blacksquare$$

Remarks. It is worthy to mention that the first of the results formulated in Theorem 3.28 was implicitly proven by Pferschy, Rudolf and Woeginger in [143]. Namely, the so called *balanced k-cut problem* formulated in [143] and solved in the case of Monge matrices is equivalent to the taxonomy problem considered in point (a) of Theorem 3.28.

Moreover, it can be shown that both f being nondecreasing and A being a lower-right graded matrix are essential conditions for point (b) of Theorem 3.28. That is, there exist examples where the violation of only one of these conditions is sufficient for the identity permutation to be a non-optimal solution to the corresponding QAP.

Example 3.2 The conditions of Theorem 3.28 are essential.
Indeed, let $n = 3$ and let the matrices A_1, A_2, B_1, B_2 be given as:

$$A_1 = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} \quad B_1 = \begin{pmatrix} 3 & 3 & 3 \\ 3 & 2 & 2 \\ 3 & 2 & 1 \end{pmatrix} \quad A_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 3 & 3 \\ 0 & 3 & 3 \end{pmatrix} \quad B_2 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 2 & 3 \\ 3 & 3 & 3 \end{pmatrix}$$

It is easily seen that A_1 is a taxonomy matrix of type $(f, 1, 2)$, where $f(1) = 3$, $f(2) = 1$, thus f is a decreasing function. Similarly, A_2 is also a taxonomy matrix and the corresponding function is nondecreasing. Additionally, B_1 is a right-lower graded Monge matrix and B_2 is a not right-lower graded matrix. The reader can check that $Z(A_1, B_1, id) = 16$, $Z(A_1, B_1, \phi) = 14$, $Z(A_2, B_2, id) = 34$ and $Z(A_2, B_2, \phi) = 24$. The last inequalities show that id is a non-optimal solution for both $QAP(A_1, B_1)$ and $QAP(A_2, B_2)$. ■

Next, notice that the condition $n_1 \leq n_2 \leq \ldots \leq n_k$ in Theorem 3.28 can be omitted. That is, for $A \in$ MONGE(LHG) or $A \in$ MONGE(RLG) and a taxonomy matrix B of type $(f, n_1, n_2, \ldots, n_k)$ with nondecreasing function f, QAP(A,B) is polynomially solvable. Indeed, given an arbitrary taxonomy matrix B of type $(f, n_1, n_2, \ldots, n_k)$, there exists a permutation ψ such that the permuted matrix B^{ψ} is again a taxonomy matrix of type $(f, n'_1, n'_2, \ldots, n'_k)$, with $n'_1 \leq n'_2 \leq \ldots \leq n'_k$. For example, for the 6×6 taxonomy matrix B of type $(f, 3, 1, 2)$, the permutation $\psi = \langle 4, 5, 6, 1, 2, 3 \rangle$ does the job. So, we get the following straightforward corollary:

Corollary 3.29 *Let A be a Monge matrix and let B be a taxonomy matrix of type $(f, n_1, n_2, \ldots, n_k)$. Assume additionally that A is either a left-higher or a right-lower graded matrix and that f is a nondecreasing function. Then QAP(A,B) is a constant permutation QAP. In the case that $A \in$ MONGE(RLG) the constant permutation is ψ and in the case that $A \in$ MONGE(LHG) the constant permutation is $\phi \circ \psi$, where $\phi = \langle n, n-1, \ldots, 2, 1 \rangle$ and ψ is the above described permutation.*

Proof. From Theorem 3.28 follows that $QAP(A, B^{\psi})$ is a constant permutation QAP. Its constant permutation is id or ϕ in the cases that $A \in$ MONGE(RLG) or $A \in$

Monge(lhG), respectively. Considering that the following equality holds for all $\pi \in \mathcal{S}_n$, $n = \sum_{i=1}^{k} n_i$, completes the proof:

$$Z(A, B, \pi \circ \psi^{-1}) = Z(A, B^{\psi}, \pi) \, . \qquad \blacksquare$$

3.6 QAPs with an Anti-Monge and a Toeplitz matrix

In this section we identify several polynomially solvable cases of a restricted version of the QAP, where one of the coefficient matrices is a left-higher graded Anti-Monge matrix and the other is a symmetric Toeplitz matrix. In Section 3.7 it will be shown that this problem is $\mathcal{NP}$-hard in general.

Moreover, we identify several conditions on the Toeplitz matrix B which lead to a simple solution for Anti-Monge(lhG)×Toeplitz(Sym). It will turn out that under each of these conditions Anti-Monge(lhG)×Toeplitz(Sym) is a constant permutation QAP.

3.6.1 Benevolent Toeplitz matrices

Let us first give the definition of the so called *benevolent functions* and *benevolent Toeplitz matrices.*

Definition 3.7 *A function* $f\colon \{-n+1, \ldots, n-1\} \to \mathbb{R}$ *is called* benevolent *if it fulfills the following three properties.*

(Ben1) $f(-i) = f(i)$ *for all* $1 \le i \le n-1$.

(Ben2) $f(i) \le f(i+1)$ *for all* $1 \le i \le \lfloor \frac{n}{2} \rfloor - 1$.

(Ben3) $f(i) \le f(n-i)$ *for all* $1 \le i \le \lceil \frac{n}{2} \rceil - 1$.

A is called a benevolent Toeplitz matrix *if it is a Toeplitz matrix generated by a benevolent function.*

Example 3.3 Let us give an example of a benevolent function f on $\{-7, -6, \ldots, 0, 1, \ldots 7\}$ defined by

$$f(0) = 1.5, \quad f(1) = 0.5, \quad f(2) = 1, \quad f(3) = 1.5, \quad f(4) = 2,$$

$$f(5) = 2.5, \quad f(6) = 2.5, \quad f(7) = 1.5$$

The graph of this function is shown in Figure 3.3. Note that the graph of this function for $x \in \{5, 6, 7\} \cup \{-5, -6, -7\}$ lies above the thin lines. This shows that in this case property Ben3 is satisfied as strict inequality. $\blacksquare$

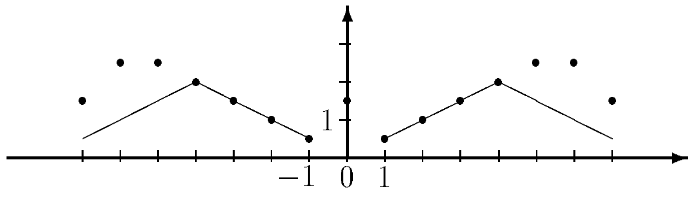


Figure 3.3: The graph of the benevolent function f

By property (Ben1), a benevolent Toeplitz matrix is symmetric.

Let us recall that permutation $\pi^* \in \mathcal{S}_n$ is defined by $\pi^*(i) = 2i-1$ for $1 \le i \le \lceil \frac{n}{2} \rceil$ and $\pi^*(n+1-i) = 2i$ for $1 \le i \le \lfloor \frac{n}{2} \rfloor$.

In this subsection we show that QAP(A,B) with $A \in$ Anti-Monge(lhG) and $B \in$ Toeplitz generated by a benevolent function, is a constant permutation QAP.

Theorem 3.30 *The permutation π^* solves the QAP(A, B) when A is a left-higher graded Anti-Monge matrix and B is a symmetric Toeplitz matrix which is generated by a benevolent function.*

Without loss of generality, we assume that all entries in the matrices A and B are nonnegative. Otherwise, we may add a sufficiently large constant to all entries without changing the combinatorial structure of QAP(A, B). All matrices in this subsection will be of dimension $n \times n$, for some fixed $n \ge 3$. If all diagonal entries of matrix B are equal to a certain constant, the value of this constant does not influence the optimal solution of QAP(A, B) but only its optimal value. Thus, since we are investigating problems of the class Anti-Monge(lhG)$\times$Toeplitz, we may assume that all Toeplitz matrices in this subsection have 0-entries on the diagonal. Consequently, $f(0) = 0$ holds for any function f generating some Toeplitz matrix in this subsection.

We want to proof that π^* solves QAP(A, B) when A and B are the specially structured matrices discussed above. We will actually show a stronger statement. We consider the relaxation RQAP(A,B) of QAP(A, B) where the columns and the rows of matrix A may be permuted *independently of each other*, by different permutations:

$$\min_{\pi,\psi \in S_n} \quad \sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i)\psi(j)} b_{ij} \tag{3.27}$$

Even for the relaxed problem RQAP(A,B), it will turn out that the expression (3.27) is minimized for $\pi = \psi = \pi^*$. This guarantees that π^* is also an optimal solution of QAP(A, B). So, instead of proving Theorem 3.30 we prove the following theorem:

Theorem 3.31 *The pair of permutations (π^*, π^*) solves RQAP(A, B) when A is a non-negative left-higher graded Anti-Monge matrix and B is a symmetric Toeplitz matrix generated by a non-negative benevolent function f with $f(0) = 0$.*

We have already seen that the set of non-negative left-higher graded Anti-Monge matrices A is a cone. Similarly, the benevolent Toeplitz matrices B form a cone, too. Clearly, (π^*, π^*) must solve $RQAP(A, B)$ when A and B are on the extremal rays of their respective cones. On the other hand, proving optimality of (π^*, π^*) for the extremal rays is also sufficient to prove the general result. This is easy to see by the following observation.

Observation 3.32 *Assume that $RQAP(A_1, B)$ and $RQAP(A_2, B)$ are both solved by the pair of permutations (π_0, ψ_0). Then for any two reals $k_1, k_2 \geq 0$, the problem $RQAP(k_1A_1 + k_2A_2, B)$ is also solved by (π_0, ψ_0).*

Proof. This follows from the equation

$$Z(k_1A_1 + k_2A_2, B, \phi, \psi) = k_1Z(A_1, B, \phi, \psi) + k_2Z(A_2, B, \phi, \psi). \qquad \blacksquare$$

Similarly, it can be proved that an analogous statement holds for the linear combinations of the second matrix B.

As we have seen in Section 3.2, the extremal rays of the set of nonnegative monotone Anti-Monge matrices have a simple structure. In the following we will see that the same holds for the extremal rays of the cone of benevolent Toeplitz matrices B. Then, we will show that π^* is optimal for every QAP(A,B) with matrices A and B belonging to the extremal rays of the respective cones. This would complete the proof of Theorem 3.31 and therefore also of Theorem 3.30.

The extremal rays of benevolent Toeplitz matrices

We will show that all nonnegative benevolent functions can be represented as nonnegative linear combinations of special benevolent functions, which only take 0-1-values. These special benevolent functions are introduced below,

Definition 3.8 *For $\lfloor \frac{n}{2} \rfloor + 1 \leq \alpha \leq n-1$, define a function $g^\alpha\colon \{-n+1, \ldots, n-1\} \to \{0,1\}$ by*

$$g^\alpha(x) = \begin{cases} 1 & \text{for } x \in \{-\alpha, \alpha\} \\ 0 & \text{for } x \notin \{-\alpha, \alpha\}. \end{cases}$$

For $1 \leq \beta \leq \lfloor \frac{n}{2} \rfloor$, define a function $h^\beta\colon \{-n+1, \ldots, n-1\} \to \{0,1\}$ by

$$h^\beta(x) = \begin{cases} 1 & \text{for } \beta \leq |x| \leq n - \beta \\ 0 & \text{otherwise.} \end{cases}$$

Example 3.4 For concreteness let us give some examples: For $n = 8$ and $\alpha = 5$ the function $g^5\colon \{-7, -6, \ldots, 0, 1, \ldots, 7\} \to \{0,1\}$ is defined by $g^5(5) = g^5(-5) = 1$ and $g^5(x) = 0$, for $x \notin \{-5, 5\}$.

For $n = 8$ and $\beta = 2$ the function $h^2\colon \{-7, -6, \ldots, 0, 1, \ldots, 7\} \to \{0,1\}$ is defined by $h^2(0) = h^2(1) = h^2(-1) = 0$, $h^2(7) = h^2(-7) = 0$ and $h^2(x) = 1$, for $x \in$

$\{2,3,4,5,6\} \cup \{-2,-3,-4,-5,-6\}$. The two Toeplitz matrices generated by these functions are

$$B(g^5) = \begin{pmatrix} 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 1&0&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0 \end{pmatrix}, \qquad B(h^2) = \begin{pmatrix} 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \\ 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \end{pmatrix}.$$

Observe the circular structure of the second matrix, which is typical of the functions h^β: Going from one row to the next corresponds to a circular right shift, and this remains true when returning from the last row to the first row. The same holds for the columns. ■

Consider the set of real functions defined on $\{-n+1, \ldots, n-1\}$ with the traditional operations "addition" and "scalar product". That is, if $f, g\colon \{-n+1, \ldots, n-1\} \to \mathbb{R}$ and $\alpha \in \mathbb{R}$, then $f+g$ and αf are defined by $(f+g)(x) = f(x)+g(x)$ and $(\alpha f)(x) = \alpha f(x)$, for $x \in \{-n+1, \ldots, n-1\}$. This set of functions with these two operations forms a linear space. The following lemma states that the subset consisting of the benevolent functions which map 0 to 0 forms a cone in this space.

Lemma 3.33 *The nonnegative benevolent functions $f\colon \{-n+1, \ldots, n-1\} \to \mathbb{R}$ with $f(0) = 0$ form a cone. The extremal rays of this cone are generated by the functions g^α and h^β.*

Proof. Since nonnegative benevolent functions are defined by linear equations and inequalities with righthand side equal to zero, they form a cone. The functions g^α and h^β are clearly benevolent. In fact, each of these functions satisfies precisely one of the characterizing inequalities (Ben2), (Ben3), and $f(1) \geq 0$ as a strict inequality and the remaining ones with equality.

We have to show that an arbitrary nonnegative benevolent function f with $f(0) = 0$ is a nonnegative linear combination of functions g^α and h^β. For this purpose we define two auxiliary functions

$$f_1(i) = \begin{cases} f(i) & \text{for} \quad |i| \leq \lfloor \frac{n}{2} \rfloor \\ f(n-i) & \text{for} \quad |i| > \lfloor \frac{n}{2} \rfloor \end{cases}$$

and

$$f_2(i) = \begin{cases} 0 & \text{for} \quad |i| \leq \lfloor \frac{n}{2} \rfloor \\ f(i) - f(n-i) & \text{for} \quad |i| > \lfloor \frac{n}{2} \rfloor \end{cases}$$

It is easily seen that $f(i) = f_1(i) + f_2(i)$ holds for all i in $\{-n+1, \ldots, n-1\}$. Finally, observe that

$$f_1 = \sum_{i=1}^{\lfloor n/2 \rfloor} [f(i) - f(i-1)] \cdot h^i \qquad\qquad f_2 = \sum_{i=\lfloor n/2 \rfloor + 1}^{n-1} [f(i) - f(n-i)] \cdot g^i \qquad (3.28)$$

and apply conditions (Ben2) and (Ben3) to see that all coefficients in these expressions are nonnegative. Hence, both f_1 and f_2 are nonnegative linear combinations of functions g^α and h^β and this completes the proof. ∎

The QAP for the special matrices $C^{(pq)}$

To complete the proof of Theorem 3.31 and hence of Theorem 3.30, we have to show that (π^*, π^*) solves $RQAP(A, B)$ when A is one of the matrices $A = C^{(pq)}$ and B is a Toeplitz matrix generated by one of the functions g_α or h_β.

Let us first specialize A to one of the matrices $C^{(pq)}$ and keep B general. Then $RQAP(C^{(pq)}, B)$ can be reformulated in a more convenient way as follows: We have to select p rows and q columns of the matrix B such that the total sum of all pq selected entries is minimized. The rows to be selected are those which are mapped to the last p rows $n-p+1, \ldots, n$ of A by permutation π, and the selected columns are those which are mapped to the last q columns of A by permutation ψ.

Lemma 3.34 (The optimal selection lemma) *Let $1 \le p, q \le n$. Let B be a nonnegative Toeplitz matrix generated by a benevolent function f with $f(0) = 0$. Suppose that p rows and q columns of the matrix B have to be selected such that the total sum of all pq selected entries of B is minimized. Then it is optimal to select the last p elements of the sequence*

$$1,\ n,\ 2,\ n-1,\ 3,\ \ldots \tag{3.29}$$

as row indices and the last q elements of this sequence as column indices. ∎

To see that Theorem 3.31 (and hence Theorem 3.30) follows from this statement note that row i of $A = C^{(pq)}$ is mapped to row $\pi^{-1}(i)$ of B. Thus the rows with the one-entries in A are mapped to the last p elements of the sequence

$$\pi^{-1}(1),\ \pi^{-1}(2), \ldots,\ \pi^{-1}(n).$$

For $\phi = \pi^*$ this sequence coincides with the sequence given in (3.29), and hence permuting the rows according to π^* in $RQAP(A, B)$ will be optimal. The same argument applies to the columns.

By Observation 3.32, this result carries over from $R^{(pq)}$ to all monotone Anti-Monge matrices A and this concludes the proof of Theorem 3.31 (hence of Theorem 3.30).

For definiteness, let us write down the rows and columns to be selected according to the above lemma. The claimed optimal solution selects the p rows from

$$p_1 := \left\lceil \frac{n-p}{2} \right\rceil + 1 \quad \text{to} \quad p_2 := n - \left\lfloor \frac{n-p}{2} \right\rfloor \tag{3.30}$$

and the q columns from

$$q_1 := \left\lceil \frac{n-q}{2} \right\rceil + 1 \quad \text{to} \quad q_2 := n - \left\lfloor \frac{n-q}{2} \right\rfloor . \tag{3.31}$$

The optimal selection lemma (Lemma 3.34) will now be proved separately for Toeplitz matrices B generated by the functions g^α (Lemma 3.35) and h^β (Lemma 3.37). Since the above formulation as a "selection" problem for rows and columns is merely another formulation of the RQAP, Observation 3.32 can then be used to obtain Lemma 3.34.

Lemma 3.35 *The pair of permutations (π^*, π^*) solves RQAP(A, B) with $A = C^{(pq)}$ and the symmetric Toeplitz matrix B generated by g^α, for any $1 \le p, q \le n$ and any $\lfloor \frac{n}{2} \rfloor + 1 \le \alpha \le n-1$.*

In other words, selecting p rows and q columns from B according to Lemma 3.34 will minimize the sum of the selected elements.

Proof. We first show that the value of the objective function must be at least $p+q-2\alpha$. Then we show that our claimed solution achieves the value $\max\{0, p+q-2\alpha\}$.

Instead of selecting p rows and q columns of the matrix B, let us take the opposite view and delete $n-p$ rows and $n-q$ columns from B. The matrix B contains originally $2(n-\alpha)$ one-entries and no two of them are in the same row or in the same column. Thus, by deleting $n-p$ rows and $n-q$ columns we may delete at most $(n-p)+(n-q)$ entries equal to 1, which gives a simple lower bound of $2(n-\alpha) - ((n-p)+(n-q)) = p+q-2\alpha$ for the remaining number of one-entries.

Let us now compute the objective function for our claimed optimal solution. We have selected columns with indices in the interval $[q_1, q_2]$, where q_1 and q_2 are given by (3.31). For some $1 \le j \le n$, column j contains at most one 1-entry, namely in row $j+\alpha$ or $j-\alpha$, whenever this index falls in the range $[1, n]$. (At most one of the two cases can occur for a given j.) Thus the selected columns contain 1-entries in the rows with indices in the set

$$([q_1-\alpha, q_2-\alpha] \cup [q_1+\alpha, q_2+\alpha]) \cap [1, n]$$

We have to intersect this set with the interval $[p_1, p_2]$ of selected rows given by (3.30) to get the number of selected 1-entries, i.e., the value of the claimed solution:

$$\Big| ([q_1-\alpha, q_2-\alpha] \cup [q_1+\alpha, q_2+\alpha]) \cap [p_1, p_2] \Big| =$$

$$\Big| ([q_1-\alpha, q_2-\alpha] \cap [p_1, p_2]) \cup ([q_1+\alpha, q_2+\alpha] \cap [p_1, p_2]) \Big|$$

Since $p_1, q_1 \le \frac{n+1}{2} \le p_2, q_2$ and $\alpha \ge \frac{n+1}{2}$, the two sets in the last union are disjoint and we may add their cardinalities. Moreover, we know the endpoints of the two intersections in the case that they are non-empty:

$$|[q_1-\alpha, q_2-\alpha] \cap [p_1, p_2]| = \max\{0,\ q_2-\alpha-p_1+1\} = \max\{0,\ \left\lfloor \frac{p}{2} \right\rfloor + \left\lceil \frac{q}{2} \right\rceil - \alpha\}$$

and

$$|[q_1+\alpha, q_2+\alpha] \cap [p_1, p_2]| = \max\{0,\ p_2-(q_1+\alpha)+1\} = \max\{0,\ \left\lceil \frac{p}{2} \right\rceil + \left\lfloor \frac{q}{2} \right\rfloor - \alpha\}.$$

For $p+q \leq 2\alpha$ both expressions are 0. For $p+q \geq 2\alpha$ the two expressions that have to be compared with 0 are non-negative and this yields

$$(\lfloor\frac{p}{2}\rfloor + \lceil\frac{q}{2}\rceil - \alpha) + (\lceil\frac{p}{2}\rceil + \lfloor\frac{q}{2}\rfloor - \alpha) = p + q - 2\alpha. \qquad \blacksquare$$

Now let us turn to the other extremal rays of the benevolent functions, the functions h^β. For the proof it will be convenient to work with a certain quantity $Q(a,m)$. We define it here and list some properties of it.

Definition 3.9 *For two integers $a, m \geq 0$, the quantity $Q(a,m)$ denotes the sum of the first a terms of the following sum:*

$$\min\{1,m\} + \min\{1,m\} + \min\{2,m\} + \min\{2,m\} + \min\{3,m\} + \min\{3,m\} + \cdots.$$

Lemma 3.36 *For fixed m, the difference $Q(a+1,m) - Q(a,m)$ increases with a. Therefore the function $Q(a,m)$ is a convex function in a with $Q(0,m) = 0$, and so we have in particular*

$$Q(a_1,m) + Q(a_2,m) \leq Q(a_1+a_2,m). \tag{3.32}$$

We also have

$$Q(a+1,m) \leq Q(a,m) + m. \tag{3.33}$$

An explicit representation of $Q(a,m)$ is

$$Q(a,m) = \begin{cases} \left\lfloor \left(\frac{a+1}{2}\right)^2 \right\rfloor & \text{for } a < 2m-2 \\ m(a+1-m) & \text{for } a \geq 2m-2 \end{cases}$$

Proof. Only the last expression is not obvious, but it can be checked by elementary calculations, which we omit. $\blacksquare$

Lemma 3.37 *The pair of permutations (π^*, π^*) solves RQAP(A,B) with $A = C^{(pq)}$ and B being a symmetric Toeplitz matrix generated by h^β, for any $1 \leq p,q \leq n$ and any $1 \leq \beta \leq \lfloor\frac{n}{2}\rfloor$.*

In other words, selecting p rows and q columns from B according to Lemma 3.34 will minimize the sum of the selected elements.

Proof. Suppose we have selected a certain set of q columns. Let x_i, $i = 1,\ldots,n$, denote the sum of the entries in the i-th row in the selected columns. Clearly, in order to minimize the sum of the entries, we have to select the p rows with the smallest x_i values.

We shall state some simple conditions on the sequence of numbers (x_i), and from this we will derive lower bounds for the sum of the p smallest numbers x_i. The lower bounds hold for any choice of q columns, and thus they provide a bound on the optimum value z^* of the objective function. This lower bound matches the objective function value of our proposed solution, and therefore this solution is optimal.

Each column of B has a very simple structure. If we arrange the row indices in a circular sequence $1, 2, \ldots, n, 1, 2, \ldots$, the one-entries in column j form a single circular interval of length $n - 2\beta + 1 =: \gamma$. This circular interval of 1s starts at row $j + \beta$ and ends at row $j + n - \beta$, wrapping around from row n to row 1 if necessary. (All indices are taken modulo n.)

If we now select q columns, the row sums x_i clearly fulfill the following condition, for every i.

$$0 \le x_i \le q. \tag{3.34}$$

It is also easy to see that, for every i,

$$q + \gamma - n \le x_i \le \gamma. \tag{3.35}$$

This upper bound follows because γ is the row sum of the whole matrix. Similarly, the lower bound $q + \gamma - n$ can be seen, because we remove $n - q$ columns from the whole matrix. We also have

$$\sum_{i=1}^{n} x_i = q\gamma, \tag{3.36}$$

because each of the q columns contains γ 1-entries.

Moreover, we have

$$|x_i - x_{i-1}| \le 1, \quad \text{for } i = 1, \ldots, n. \tag{3.37}$$

(In this section, all indices are taken modulo n, and so x_0 denotes the same variable as x_n.) This inequality follows from the fact that there is only one column which has a 0 in row $i - 1$ and a 1 in row i. Analogously, there is only one column with a 1 in row $i - 1$ and and 0 in row i, for every i.

The constraints (3.34) and (3.35) imply

$$p \cdot \max\{0, q + \gamma - n\} \le z^* \le p \cdot \min\{q, \gamma\}.$$

We will first deal with the cases in which these simple bounds are sufficient. Let us look at the vector x_i for our proposed optimal solution. Figure 3.4 shows such a sequence for the case $n = 19$, $q = 5$ and $\gamma = 8$. Since we select q "adjacent" columns, the circular sequence of values x_i consists of two horizontal intervals connected by a rising and a falling interval of slope ± 1. By analyzing the situation in detail, one can make the following statements about the two horizontal pieces. The higher piece looks as follows:

- Case U1. $q \ge \gamma$. The maximum value of x_i is γ and it occurs for $q - \gamma + 1$ adjacent positions (rows).

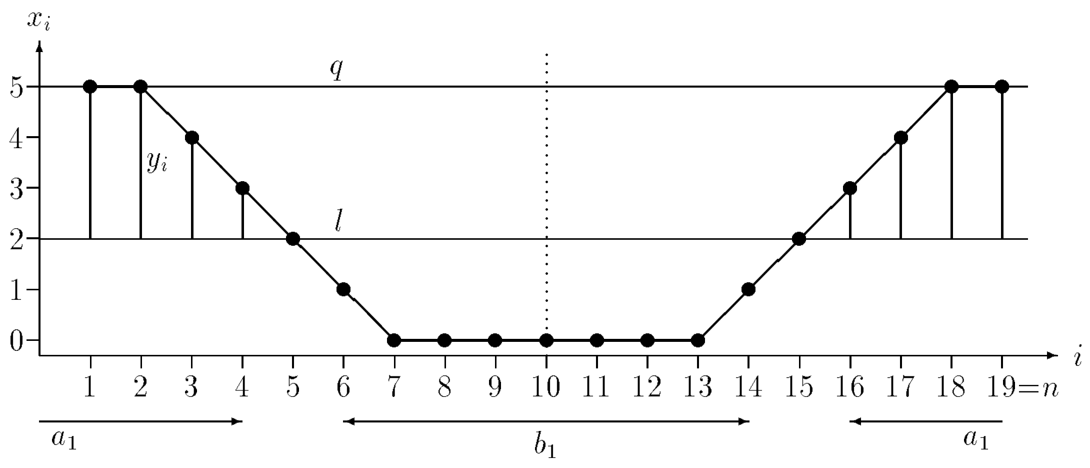


Figure 3.4: The vector (x_i) for the optimal selection of columns for an example with $n = 19$, $q = 5$, and $\gamma = 8$ ($\beta = 6$). The indices of the selected columns are 8, 9, 10, 11 and 12. If $p = 10$ or 11, the p-smallest x_i has the value $l = 2$ indicated by the horizontal line.

- Case U2. $q \leq \gamma$. The maximum value of x_i is q and it occurs for $\gamma - q + 1$ adjacent positions.

(The two cases are not mutually exclusive.) The lower piece looks as follows:

- Case L1. $q + \gamma \leq n$. The minimum value of x_i is 0 and it occurs for $n - q - \gamma + 1$ adjacent positions.
- Case L2. $q + \gamma \geq n$. The minimum value of x_i is $q + \gamma - n$, and it occurs for $q + \gamma - n + 1$ adjacent positions.

These maximum and minimum values of x_i coincide with the ones given by (3.34) and (3.35). One can also check that in the sequence given by (3.29), the first rows selected (i.e., the last elements of the sequence) correspond to the positions where x_i has the minima, and the last rows selected correspond to the positions where x_i has its maxima. Therefore, if p has such a small value that only the minimal positions are selected, we know that the solution is optimal.

So we get the solution for the following cases (which are not mutually exclusive):

- Case L1 ($q + \gamma \leq n$): If $p \leq n - q - \gamma + 1$, then

$$z^* = 0. \tag{3.38}$$

- Case L2 ($q + \gamma \geq n$): If $p \leq q + \gamma - n + 1$, then

$$z^* = p(q + \gamma - n). \tag{3.39}$$

Similarly, if p is big enough so that all elements except possibly the largest elements are selected, the solution is guaranteed to be optimal, taking into account that the total sum of all entries is constant, by (3.36).

So we get the solutions for the following additional cases:

- Case U1 ($q \geq \gamma$): If $p \geq n-(q-\gamma+1)$, then
$$z^* = q\gamma-(n-p)\gamma = (p+q-n)\gamma \tag{3.40}$$
- Case U2 ($q \leq \gamma$): If $p \geq n-(\gamma-q+1)$, then
$$z^* = q\gamma-(n-p)q = (p+\gamma-n)q. \tag{3.41}$$

The remaining case that has to be considered is
$$|n-q-\gamma|+1 < p < n-|q-\gamma|-1. \tag{3.42}$$

In this case one can also check that the p rows which are selected in our proposed optimal solution correspond to the p smallest x_i-values, if the q columns have been selected according to Lemma 3.34. Thus the objective function value z^* corresponding to the solution claimed to be optimal is evaluated as follows:

- Case L1 ($q+\gamma \leq n$): $z^* = (n-q-\gamma+1)\times 0+[1+1+2+2+3+3+\cdots]$, where $p-(n-q-\gamma+1)$ numbers are taken from the second sum. Now Lemma 3.36 yields
$$z^* = Q(p+q+\gamma-n-1,\infty) = \left\lfloor \left(\frac{p+q+\gamma-n}{2}\right)^2 \right\rfloor. \tag{3.43}$$
- Case L2 ($q+\gamma \geq n$): According to the above description of the vector x_i, every x_i is at least $q+\gamma-n$, and this value is taken by $q+\gamma-n+1$ elements x_i. Summing separately the excess of x_i over $q+\gamma-n$, we can write
$$z^* = p\times(q+\gamma-n)+(q+\gamma-n+1)\times 0+[1+1+2+2+3+3+\cdots],$$
where $p-(q+\gamma-n+1)$ numbers are taken from the sum in brackets. This yields
$$\begin{aligned} z^* &= p\times(q+\gamma-n)+Q(p-(q+\gamma-n+1),\infty) \\ &= \left\lfloor \frac{4p(q+\gamma-n)+(p-(q+\gamma-n))^2}{4} \right\rfloor = \left\lfloor \frac{(p+q+\gamma-n)^2}{4} \right\rfloor, \end{aligned}$$
which coincides with (3.43).

To prove that this is a lower bound we also have to take into account inequalities (3.37).

Let l denote the p-smallest element among the n elements x_i, which is at the same time the $(n-p+1)$-largest element, and let l' denote the $(p+1)$-smallest (the $(n-p)$-largest) among the elements x_i. Let z_{small} denote the sum of the p smallest x_i values, and let $I_{\text{small}} \subseteq \{1,\dots,n\}$ denote the index set of the p smallest x_i-values. (This set is not uniquely defined if several x_i-s are equal to l. We can resolve ties arbitrarily.) Let $I_{\text{large}} = \{1,\dots,n\} - I_{\text{small}}$ the complementary index set of the $n-p$ largest x_i values and let $z_{\text{large}} = q\gamma - z_{\text{small}}$ denote the sum of these values. We would like to bound z_{small} from below, or, what amounts to the same thing, to bound z_{large} from above.

Let us look at a maximal block of consecutive elements x_i in the circular sequence which are larger than l:

$$x_{i_0} = l, \quad x_{i_0+1}, x_{i_0+2}, \dots, x_{i_0+a_j} > l, \quad x_{i_0+a_j+1} = l \tag{3.44}$$

(Recall that all indices are interpreted modulo n. The block length $a_j = n-1$ is permitted.) Let $a_1, a_2, \dots, a_k$ denote the lengths of all these blocks. By definition, all these $a := a_1 + a_2 + \cdots + a_k$ elements belong to I_{large} and so we have $a \le n-p$. For $i \in I_{\text{large}}$ we substitute $y_i = x_i - l \ge 0$. These quantities are indicated by vertical lines in Figure 3.4. We have

$$z_{\text{large}} = \sum_{i \in I_{\text{large}}} x_i = (n-p)l + \sum_{i \in I_{\text{large}}} y_i.$$

Let us look at one block of length a_j as defined in (3.44). The maximum possible sum $y_{i_0+1} + y_{i_0+2} + \cdots + y_{i_0+a_j}$ of such a block can be estimated by setting up a linear program with the constraints coming from (3.34), (3.35), and (3.37):

$$\max \left\{ \sum_{i=1}^{a} \hat{y}_i \,\middle|\, \hat{y}_0 = \hat{y}_{a+1} = 0;\ \hat{y}_i \le m \text{ and } |\hat{y}_i - \hat{y}_{i-1}| \le 1 \text{ for } 1 \le i \le a+1 \right\},$$

with $m := \min\{q,\gamma\} - l$. In this linear program, the variables $\hat{y}_i$ represent the possible values for y_{i_0+i}. It is easy to see (cf. Figure 3.4) that the optimal value is equal to the first a_j terms of the sum:

$$\min\{1,m\} + \min\{1,m\} + \min\{2,m\} + \min\{2,m\} + \min\{3,m\} + \min\{3,m\} + \cdots,$$

which we defined as $Q(a_j, m)$. Thus we get

$$y_{i_0+1} + y_{i_0+2} + \cdots + y_{i_0+a_j} \le Q(a_j, \min\{q,\gamma\} - l).$$

By summing over all k blocks we get the following bound

$$z_{\text{large}} \le (n-p)l + \sum_{j=1}^{k} Q(a_j, \min\{q,\gamma\} - l). \tag{3.45}$$

By repeatedly using the relation $Q(a_1, m) + Q(a_2, m) \le Q(a_1 + a_2, m)$ from (3.32), we can simplify this to our first essential bound:

$$z_{\text{large}} \le (n-p)l + Q(a, \min\{q,\gamma\} - l) \le (n-p)l + Q(n-p, \min\{q,\gamma\} - l) \tag{3.46}$$

We can apply a similar reasoning to l' instead of l. The indices i with $x_i \ge l'$ still include all elements of I_{large}. The only difference is that for one of the $n-p$ elements $i \in I_{\text{large}}$ we must surely have $x_i = l'$, and therefore the number of x_i which are strictly larger than l' is at most $n-p-1$. So we get a second bound

$$z_{\text{large}} \le (n-p)l' + Q(a, \min\{q,\gamma\} - l') \le (n-p)l' + Q(n-p-1, \min\{q,\gamma\} - l'). \tag{3.47}$$

Now we apply analogous considerations to z_{small}. For the p elements $i \in I_{\text{small}}$ we introduce the nonnegative quantities $y_i = l - x_i$ or $y_i = l' - x_i$, respectively, and their sum can be bounded in terms of the Q function. The roles of l and l' are now reversed: The number of x_i which are strictly smaller than l is at most $p-1$ and the number of x_i which are strictly smaller than l' is at most p. This gives the following bounds.

$$\begin{aligned} z_{\text{small}} &\ge pl' - Q(p, l' - \max\{0, q+\gamma-n\}) && (3.48)\\ z_{\text{small}} &\ge pl - Q(p-1, l - \max\{0, q+\gamma-n\}) && (3.49) \end{aligned}$$

Using the relation $z_{\text{small}} = q\gamma - z_{\text{large}}$ we thus get the following four lower bounds on z^*.

$$\begin{aligned} U_1^-(l) &= pl - Q(p-1, l - \max\{0, q+\gamma-n\}) && (3.50)\\ U_2^-(l') &= pl' - Q(p, l' - \max\{0, q+\gamma-n\}) && (3.51)\\ U_1^+(l') &= q\gamma - (n-p)l' - Q(n-p-1, \min\{q,\gamma\} - l') && (3.52)\\ U_2^+(l) &= q\gamma - (n-p)l - Q(n-p, \min\{q,\gamma\} - l) && (3.53) \end{aligned}$$

These bounds may be combined into one lower bound, which depends on the quantities l and l'. Since these are unknown, we have to minimize over all choices of l and l', subject only to the constraints $l \le l' \le l+1$. The inequality $l' \le l+1$ follows from (3.37). Thus,

$$z^* \ge \min_{\substack{l \le l' \le l+1 \\ \max\{0, q+\gamma-n\} \le l \le l' \le \min\{q,\gamma\}}} \max\{U^-(l,l'), U^+(l,l')\}, \tag{3.54}$$

with

$$U^-(l,l') = \max\{U_1^-(l), U_2^-(l')\} \quad \text{and} \quad U^+(l,l') = \max\{U_1^+(l'), U_2^+(l)\}.$$

From $Q(a+1, m) \le Q(a, m) + m$ (see (3.33)) it follows that $U_1^-(l)$, $U_2^-(l')$, and hence $U^-(l,l')$, are nondecreasing in l and in l'. Similarly $U^+(l,l')$ is nonincreasing in l and l'. Thus, if we consider the possible pairs (l,l') in the order

$$(0,0),\ (0,1),\ (1,1),\ (1,2),\ (2,2),\ (2,3),\ (3,3), \ldots,$$

then U^- increases and U^+ decreases. It suffices therefore to exhibit a pair (l, l') for which $U^-(l, l') = U^+(l, l')$ in order to identify the point in (3.54) where the minimum occurs and to produce in this way a valid lower bound for z^*. We will show that the pair of values

$$l = \left\lfloor \frac{p+q+\gamma-n}{2} \right\rfloor \text{ and } l' = \left\lceil \frac{p+q+\gamma-n}{2} \right\rceil$$

has this property, and the resulting bound is equal to the value z^* of our proposed optimal solution computed in (3.43). This will involve some calculations. We must distinguish two cases.

Case 1. $p+q+\gamma+n \equiv 0 \ (mod\ 2)$. We have $l = l' = (p+q+\gamma-n)/2$. In this case the two expressions $U_1^-(l)$ and $U_2^-(l')$ differ only in the first argument of the function Q. Using the monotonicity of Q in its first argument we conclude that $U^-(l, l') = U_1^-(l)$ and similarly that $U^+(l, l') = U_1^+(l')$.

The expressions which occur as the second arguments to Q in (3.50) can be expressed as follows.

$$m^- := l - \max\{0, q+\gamma-n\} = \frac{p - |q+\gamma-n|}{2} \quad \text{and}$$

$$m^+ := \min\{q, \gamma\} - l' = \frac{n-p-|q-\gamma|}{2}.$$

These arguments always fall in the range where the function Q is linear. We have to check that $p - 1 \geq 2m^- - 2$ and $n - p - 1 \geq 2m^+ - 2$. These inequalities are obviously fulfilled. We can therefore use the expression $Q(a, m) = m(a + 1 - m)$ in all four cases. Substitution of l and m^- into the respective formulas (3.50) yields

$$\begin{aligned} U^-(l,l') &= U_1^-(l) = \frac{p(p+q+\gamma-n)}{2} - \frac{p-|q+\gamma-n|}{2} \cdot \left(p - \frac{p-|q+\gamma-n|}{2}\right) \\ &= \frac{p(p+q+\gamma-n)}{2} - \frac{(p-|q+\gamma-n|)\cdot(p+|q+\gamma-n|)}{4} \\ &= \frac{2p^2+2p(q+\gamma-n)-p^2+(q+\gamma-n)^2}{4} = \left(\frac{p+q+\gamma-n}{2}\right)^2 \end{aligned}$$

$U^+(l, l') = U_1^+(l')$ can be evaluated in a similar way and it yields the same result. This concludes the proof for the first case.

Case 2. $p+q+\gamma+n \equiv 1 \ (mod\ 2)$. We have $l = (p+q+\gamma-n-1)/2$ and $l' = l+1$. Again it can be checked easily that the four expressions $m^- := \frac{p-|q+\gamma-n|-1}{2}$, $m^- + 1$, $m^+ := \frac{n-p-|q-\gamma|+1}{2}$ and $m^+ - 1$, which occur as second arguments to Q in (3.50), always fall in the range where the function Q is linear.

The two expressions $U_1^-(l)$ and $U_2^-(l')$ can be compared as follows.

$$U_2^-(l') - U_1^-(l) = p - Q(p, m^- + 1) + Q(p-1, m^-)$$

Using the equation $Q(p, m^-+1)-Q(p-1, m^-) = p-m$ we conclude that this difference is nonnegative and hence $U^-(l, l') = U_2^-(l')$. Similarly, we have $U^+(l, l') = U_2^+(l)$. Both expressions can be evaluated just as in Case 1 and yield the same result:

$$U^-(l, l') = U^+(l, l') = \frac{(p+q+\gamma-n)^2-1}{4}$$

This concludes the proof for the second case. Since the lower bound for z^* which we have just proved coincides with the value given in (3.43) the proof of the lemma is complete. ■

A different proof of Lemma 3.37 can be given by using a result of Çela and Woeginger (Theorem 4.1 in [37]). They showed, translated into the language of Lemma 3.34, that there exists always an optimal selection consisting of a block of p (cyclically) adjacent rows and of a block of q adjacent columns. Their result is formulated in a graph-theoretic setting, and the proof is based on an exchange argument.

We remark that formula (3.43) can be combined with the previous bounds (3.38)–(3.41) for the easy cases L1, L2, U1 and U2 into one closed-form expression for the optimum objective function value. Let $N := \max\{0,\, p+q+\gamma-n\}$ and $k := \min\{p, q, \gamma, \lfloor N/2 \rfloor\}$. Then

$$z^* = N \cdot (N-k).$$

It is perhaps astonishing that this formula is completely symmetric in p, q, and γ.

3.6.2 Periodic Toeplitz matrices

In this subsection we derive the second polynomially solvable case of the problem:

$$\text{Anti-Monge(lhG)}\times\text{Toeplitz(Sym)} \qquad (3.55)$$

We consider periodic Toeplitz matrices, i.e., Toeplitz matrices whose generating function is periodic. We define a generalized benevolent function, the so called k-benevolent function and show that the problem in (3.55) with a Toeplitz matrix generated by such a function is polynomially solvable. Moreover, we show that for general periodic Toeplitz matrices the problem in (3.55) is $\mathcal{NP}$-hard.

Toeplitz matrices generated by k-benevolent functions

Here we extend the benevolent functions in a periodic way and we show that for the resulting Toeplitz matrices the QAP in (3.55) is a constant permutation QAP.

Definition 3.10 *Let $k \geq 1$ and $n = kn'$. A function $f\colon \{-n+1, \ldots, n-1\} \to \mathbb{R}$ is called k-*benevolent *if it fulfills the following four properties.*

(i) $f(i) \leq f(i+1)$, *for* $0 \leq i \leq \lfloor \frac{n'}{2} \rfloor - 1$.

(ii) $f(i) = f(n' - i)$, *for* $0 \le i \le \lceil \frac{n'}{2} \rceil - 1$.

(iii) $f(i) = f(i + jn')$, *for* $0 \le i \le n' - 1$, $1 \le j \le k - 1$.

(iv) $f(-i) = f(i)$, *for* $0 \le i \le n - 1$.

*A Toeplitz matrix generated by a k-benevolent function is called k-*benevolent Toeplitz matrix*.*

Properties (i), (ii), and (iv) are the same as the properties of benevolent functions for the range $\{-n' + 1, \dots, n' - 1\}$, with two exceptions: The symmetry condition (ii) requires equality, and $f(0)$ is involved in (i). Property (iv) provides the periodic continuation with period n'.

Example 3.5 Let $n = 15$, $k = 3$, $n' = 5$. Define a function $f\colon \{-14, -13, \dots, 0, 1, \dots, 14\} \to \mathbb{R}$, fulfilling properties (i)–(iv), by $f(0) = 1$, $f(1) = 2$, $f(2) = 3$. The Toeplitz matrix B generated by this 3-benevolent function is given below.

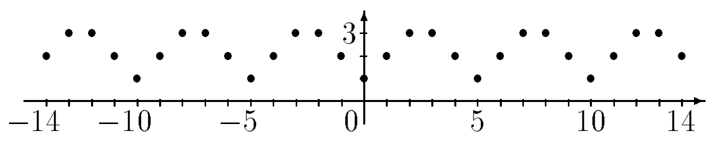


Figure 3.5: The graph of the function f in 3.5.

$$B = \left(\begin{array}{ccccc:ccccc:ccccc}
0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 \\
1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 \\
2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 \\
2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 \\
1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 \\
\hdashline
0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 \\
1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 \\
2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 \\
2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 \\
1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 \\
\hdashline
0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 \\
1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 \\
2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 \\
2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 \\
1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0 & 1 & 2 & 2 & 1 & 0
\end{array}\right)$$

One can see clearly that the matrix consists of $3 \times 3 = 9$ identical blocks of size $n' \times n' = 5 \times 5$ each. Two columns whose indices are congruent modulo $n' = 5$ are identical, and the same holds for the rows. ■

Next, we describe the permutation which is optimal for the QAP(A,B) of size $n = kn'$ with $A \in$ Anti-Monge(lhG) and a Toeplitz matrix B generated by a k-benevolent function. We recall from Definition 3.5 that $\pi^* = \langle 1, 3, 5, 7, 9, \dots, 8, 6, 4, 2\rangle$. In this subsection, π^* will be regarded as a permutation of $\{1, \dots, n'\}$. With the help of this permutation $\pi^* \in S_{n'}$, we construct the permutation $\pi^{(k)} \in S_n$ (with $n = kn'$) by the rule

$$\pi^{(k)}((u-1)n' + i) = k\pi^*(i) - (u-1), \qquad \text{for } 1 \le u \le k,\ 1 \le i \le n'. \tag{3.56}$$

For example, for $k = 4$, $n' = 5$, $n = kn' = 20$,

$$\pi^{(4)} = \langle \underbrace{4, 12, 20, 16, 8}, \underbrace{3, 11, 19, 15, 7}, \underbrace{2, 10, 18, 14, 6}, \underbrace{1, 9, 17, 13, 5} \rangle.$$

The sequence $\langle \pi^{(k)}(1), \pi^{(k)}(2), \dots \rangle$ is naturally divided into $k = 4$ groups with $n' = 5$ elements each. The first group, corresponding to $u = 1$ in (3.56), is obtained from $\pi^* = \langle 1, 3, 5, 4, 2\rangle$ by multiplying every element by $k = 4$. Each successive group is obtained from the previous one by subtracting one from each entry. Thus, the numbers in the i-th group are those numbers between 1 and $n = 20$ which are congruent to $-(i-1)$ modulo k.

Now we divide the set of indices $\{1, 2, \dots, n\}$ into k blocks. We denote the indices of block u by

$$N_u := \{(u-1)n' + 1, \dots, un'\}.$$

As mentioned above, these blocks partition the row and column indices of the matrix B in such a way that all submatrices B_{uv} whose rows are selected by some block N_u and whose columns are selected by some block N_v are identical.

Now we can state the above mentioned polynomiality result.

Theorem 3.38 *The permutation $\pi^{(k)}$ solves QAP(A, B) when A is a left-higher graded Anti-Monge matrix and B is a symmetric Toeplitz matrix which is generated by a k-benevolent function.*

As previously, we can show that for such matrices A and B, $(\pi^{(k)}, \pi^{(k)})$ solves even RQAP(A,B). We can achieve $f(0) = 0$ by subtracting a constant from all values of f. This does not change the combinatorial structure of $RQAP(A, B)$. Since $f(0)$ is the smallest value of f, the resulting matrix B will be non-negative.

Theorem 3.39 *The pair of permutations $(\pi^{(k)}, \pi^{(k)})$ solves RQAP(A, B) when A is a non-negative left-higher graded Anti-Monge matrix and B is a symmetric Toeplitz matrix which is generated by a k-benevolent function with $f(0) = 0$.*

As in Section 3.6.1, we can restrict our attention to the matrices $A = C^{(pq)}$ which are the extreme rays of the cone of non-negative left-higher graded Anti-Monge matrices. It would be easy to find the extreme rays of non-negative k-benevolent Toeplitz matrices B, but we do not need this because the proof will directly rely on some lemmas of Section 3.6.1.

Lemma 3.40 *The pair of permutations $(\pi^{(k)}, \pi^{(k)})$ solves RQAP(A, B) with $A = C^{(pq)}$ and a Toeplitz matrix B generated by a k-benevolent function with $f(0) = 0$, for any $1 \leq p, q \leq n$.*

Proof. We know that this problem can be seen as selecting p rows and q columns of matrix B such that the total sum of all pq selected entries is minimized. Now suppose that some q columns have already been selected and we have to select the rows. Let x_i for $i = 1, \ldots, n$ denote the sum of the selected entries in row i. Clearly, we have to select those p rows with the smallest x_i values. Since rows of B whose indices i are congruent modulo n' are identical, the numbers x_i corresponding to these rows are equal. Therefore, if $v_1 < v_2 < \ldots < v_j$ are the values taken by the elements x_i and $V_t = \{x_i \mid x_i = v_t\}$, then $|V_t|$ is a multiple of k, for any $1 \leq t \leq j$. Moreover, the elements of V_t "are uniformly distributed in blocks", i.e., there are $|V_t|/k$ elements of V_t belonging to each block, for $1 \leq t \leq j$. Thus we may impose the following structure on the selected set of columns.

Claim 3.41 *There is an optimal selection of p rows, where the number p_u of selected rows in each block N_u is either $\lfloor p/k \rfloor$ or $\lceil p/k \rceil$.*

Since we can equally apply the argument to the rows once the columns are selected (in accordance with Claim 3.41), we also get:

Claim 3.42 *There is an optimal selection of p rows and q columns, where, in addition to the property of Claim 3.41, the number q_v of selected columns in each block N_v is either $\lfloor q/k \rfloor$ or $\lceil q/k \rceil$.*

The entries of B which lie in the selected rows and columns can be summed separately for each block B_{uv}, $(1 \leq u, v \leq k)$. All blocks B_{uv} are identical to a certain $n' \times n'$ benevolent Toeplitz matrix B'. By Lemma 3.34 we know how to optimally select a given number p' of rows and a given number q' of columns from B' if we want to minimize the overall sum of the selected entries. Let us denote by $z(B', p', q')$ the optimal value of this problem, i.e., the value of RQAP$(C^{(p'q')}, B')$. So we get the following lower bound for our problem.

$$Z(C^{(pq)}, B, \pi, \psi) \geq \sum_{u=1}^{k} \sum_{v=1}^{k} z(B', p_u, q_v)$$

Let us denote $r_p := p \bmod k$. Then r_p of the values p_u must be equal to $\lceil p/k \rceil$ and the remaining $k - r_p$ of the values p_u are equal to $\lfloor p/k \rfloor$. Similarly, $r_q := q \bmod k$ of the values q_v are equal to $\lceil q/k \rceil$ and $k - r_q$ of them are equal to $\lfloor q/k \rfloor$. To finish the proof of the lemma, we have to show that the permutation $\pi^{(k)}$ indeed selects the optimal set of p_u rows out of each block of rows N_u and the optimal set of q_v columns out of each block of columns N_v, as specified by Lemma 3.34. This is easy to check: The selected row and column indices i are those which satisfy $\pi^{(k)}(i) > n - p$

or $\pi^{(k)}(i) > n-q$, respectively. By the way how $\pi^{(k)}$ is constructed, if we look at the indices of selected rows in each block N_u, these are precisely the positions where the p_u largest entries in π^* occur:

$$\pi^{(k)}((u-1)n'+i) > n-p \text{ if and only if } \pi^*(i) > n'-p_u, \quad \text{for } 1 \le u \le k,\ 1 \le i \le n'\ ,$$

where

$$p_u = \begin{cases} \lceil p/k \rceil & \text{for } u = 1, \ldots, r_p, \\ \lfloor p/k \rfloor & \text{for } u = r_p + 1, \ldots, k. \end{cases}$$

The same situation holds for the columns and this is just in accordance with Lemma 3.34. ■

Toeplitz matrices generated by general periodic functions

The simplest non-trivial periodic functions f for generating a Toeplitz matrix B have period $n' = 2$ and thus only two values: $f(0) = f(i)$ for all even i and $f(1) = f(i)$ for all odd i. These two values form a chess-board pattern in the matrix B. The case $f(0) \le f(1)$ was treated above. It leads to a k-benevolent function (if n is even) and hence to the constant permutation QAP. In this section we deal with the other case, $f(0) > f(1)$ and show that it represents an NP-hard problem. It is no loss of generality to assume $f(0) = 1$ and $f(1) = -1$. In this case $B = (b_{ij})$ can be written as $b_{ij} = (-1)^{i+j}$.

Theorem 3.43 *QAP(A,B) is $\mathcal{NP}$-hard even if A is a $(2k) \times (2k)$ left-higher graded Anti-Monge matrix and $B = (b_{ij})$ is a $(2k) \times (2k)$ symmetric 0-1 Toeplitz matrix with $b_{ij} = (-1)^{i+j}$.*

Proof. From $A \in$ Anti-Monge(lhG) and $B \in$ Chess, it follows that $-A \in$ Monge and $-B \in$ NegChess. In Section 3.4 we have argued that the problems $QAP(A,B)$ and $QAP(-A,-B)$ are equivalent. Table 3.1 shows that Monge$\times$NegChess is $\mathcal{NP}$-hard. Thus $QAP(-A,-B)$ is $\mathcal{NP}$-hard and this completes the proof. ■

3.6.3 Symmetric Toeplitz matrices with small bandwidth

According to Definition 3.4 the bandwidth of a Toeplitz matrix generated by function f is the smallest i such that $f(i) = 0$. In the first part of this subsection we show that symmetric Toeplitz matrices with bandwidth two lead to constant permutation QAPs. In the second part we give some examples which show that for any bandwidth larger than two, the resulting problem is no longer a constant permutation QAP.

Toeplitz matrices with bandwidth two

In this section, we investigate symmetric Toeplitz matrices that have bandwidth two. We show that QAP(A,B) with $A \in$ Anti-Monge(lhG) and a bandwidth-2 Toeplitz matrix B is a constant permutation QAP. By modifying the main diagonal we may assume that $f(0) = 0$ and hence $f(1) = f(-1)$ are the only two non-zero values of f.

The case $f(1) = f(-1) \leq 0$ leads to a benevolent function, which is covered by Theorem 3.30. So the interesting case which remains is $f(1) = f(-1) > 0$. By scaling we may then assume without loss of generality that $f(1) = f(-1) = 1$.

The optimal permutation for these matrices is the so-called zig-zag permutation.

Definition 3.11 *The* zig-zag *permutation $\pi_{\mathrm{z}} \in S_n$ defined as follows:*

$$\pi_{\mathrm{z}}(i) = \begin{cases} n-i & \text{if } i \leq \frac{n}{2} \text{ and } i \text{ is odd} \\ i & \text{if } i \leq \frac{n}{2} \text{ and } i \text{ is even} \\ i & \text{if } i > \frac{n}{2} \text{ and } n-i \text{ is even} \\ n-i & \text{if } i > \frac{n}{2} \text{ and } n-i \text{ is odd} \end{cases}$$

Example 3.6 For $n = 12$ we have $\pi_{\mathrm{z}} = \langle \underline{11}, 2, \underline{9}, 4, \underline{7}, 6, \underline{5}, 8, \underline{3}, 10, \underline{1}, 12 \rangle$ and for $n = 13$ we have $\pi_{\mathrm{z}} = \langle \underline{12}, 2, \underline{10}, 4, \underline{8}, 6, 7, \underline{5}, 9, \underline{3}, 11, \underline{1}, 13 \rangle$. The underlined entries are those which are computed as $\pi_{\mathrm{z}}(i) = n - i$ in the above formula. Note that, for even n, the distinction between the cases $i \leq \frac{n}{2}$ and $i > \frac{n}{2}$ is irrelevant. ■

As in Theorem 3.31, we may even permute the rows and columns independently. Then, the resulting RQAP and the considered QAP have the same optimal value.

Theorem 3.44 *For any $n \geq 1$, the pair of permutations $(\pi_{\mathrm{z}}, \pi_{\mathrm{z}})$ solves RQAP(A, B) for a left-higher graded Anti-Monge matrix A and the symmetric Toeplitz matrix B generated by the function $f\colon \{-n+1, \ldots, n-1\} \to \{0, 1\}$, with $f(1) = f(-1) = 1$ and $f(i) = 0$ for $i \neq \pm 1$.*

Proof. It is sufficient to show that permutation π_{z} is an optimal solution of *RQAP*(A, B) for all $A = C^{(pq)}$ with $1 \leq p, q \leq n$. We again use the formulation of RQAP(A,B) as a row and column selection problem: We have to select p rows and q columns of the matrix B such that the total sum of all pq selected entries is minimized. The claim is that it is optimal to select the rows $\pi_{\mathrm{z}}^{-1}(i)$, for $i \geq n-p+1$, and the columns $\pi_{\mathrm{z}}^{-1}(j)$, for $j \geq n-q+1$.

We will first compute the objective function value for π_{z} as

$$Z(C^{(pq)}, B, \pi_{\mathrm{z}}, \pi_{\mathrm{z}}) = \begin{cases} 0 & \text{if } p+q-n \leq 0 \\ 1 & \text{if } p+q-n = 1 \text{ and } p \neq q \\ 0 & \text{if } p+q-n = 1 \text{ and } p = q = \frac{n+1}{2} \\ 2(p+q-n-1) & \text{if } p+q-n \geq 2. \end{cases} \qquad (3.57)$$

Then it will be rather easy to show that this expression is a lower bound for $Z(C^{(pq)}, B, \pi, \psi)$.

To compute $Z(C^{(pq)}, B, \pi_Z, \pi_Z)$, recall that $b_{ij} = 1$ if $j = i+1$ or $j = i-1$ and $b_{ij} = 0$ otherwise. Thus we have a contribution of 1 to the sum

$$Z(C^{(pq)}, B, \pi_Z, \pi_Z) = \sum_{i=1}^{n}\sum_{j=1}^{n} C^{(p,q)}_{\pi_Z(i),\pi_Z(j)} b_{ij}$$

for each pair (i,j) with $j = i \pm 1$ with $\pi_Z(i) \geq n-p+1$ and $\pi_Z(j) \geq n-q+1$. For a given value $u = \pi_Z(i)$ we call the values $\pi_Z(i \pm 1)$ the two *neighbors* of u. The neighbors are the two numbers which are adjacent to u in the sequence $\langle \pi_Z(1), \pi_Z(2), \ldots \rangle$, see Example 3.6. Of course, $\pi_Z(1) = n-1$ and $\pi_Z(n) = n$ have only one neighbor each. We now distinguish two cases.

Case 1. n is even. Then the sum of two neighbors is always equal to $n-1$ or $n+1$, as can be checked from Definition 3.11. More precisely, we have:

- Every index $u \leq n-2$ has two neighbors u' and u''. For one of them, the sum $u + u' = n+1$, and for the other one, $u + u'' = n-1$.
- The indices $u = n-1$ and $u = n$ have one neighbor u' each and the sum $u + u' = n+1$.

We have to look at the numbers $u \geq n-p+1$ and see how many neighbors u' with $u' \geq n-q+1$ they have. An easy calculation shows that

- If $u \leq q-2$ (and hence $u \leq n-2$), then both neighbors $u' = n+1-u$ and $u'' = n-1-u$ fulfill $u', u'' \geq n-q+1$.
- If $u = q-1$ or $u = q$, then just one neighbor of u, namely $u' = n+1-u$, fulfills $u' \geq n-q+1$.
- If $u \geq q+2$, no neighbor u' of u fulfills $u' \geq n-q+1$.

Thus, counting for each $u = n-p+1, \ldots, n$ the number of neighbors u' with $u' \geq n-q+1$, we get

$$2 \cdot |\{n-p+1, \ldots, n\} \cap \{1, \ldots, q-2\}| + |\{n-p+1, \ldots, n\} \cap \{q-1, q\}|,$$

Which can be checked to be equal to (3.57). (The third case in (3.57) cannot occur when n is even.)

Case 2. n is odd. The situation is similar as in Case 1, with only one exception. The sum of the two neighbors $\pi_Z(\frac{n-1}{2}) = \frac{n-1}{2}$ and $\pi_Z(\frac{n+1}{2}) = \frac{n+1}{2}$ equals n, and not $n \pm 1$.

In considering the two neighbors of each index u, there are two changes:

- For $u = \frac{n-1}{2}$, we have a neighbor $u' = \frac{n+1}{2}$ (with sum $u + u' = n$), instead of the neighbor $u' = \frac{n-1}{2}$ (with sum $u + u' = n-1$).

- On the other hand for $u = \frac{n+1}{2}$, we have a new neighbor $u'' = \frac{n-1}{2}$ (with sum $u + u' = n$), instead of a neighbor $u'' = \frac{n+1}{2}$ (with sum $u + u'' = n + 1$).

The effects of these two changes cancel each other except in one special case: $p = q = \frac{n+1}{2}$. In this case, we lose one pair of neighbors and we get $Z(C^{(pq)}, B, \pi_Z, \pi_Z) = 0$ instead of the value of 1 given by the general formula.

To conclude the proof, let us show that (3.57) is a lower bound on the objective function. Instead of selecting p rows and q columns of the matrix B, let us take the opposite view and delete $n-p$ rows and $n-q$ columns from B. The matrix B originally contains $2n-2$ one-entries, and each row or column contains at most two one-entries. Thus, by deleting $n-p$ rows and $n-q$ columns we may delete at most $2(n-p)+2(n-q)$ ones, which gives an easy lower bound of $Z(C^{(pq)}, B, \pi, \psi) \geq 2n-2-(2(n-p)+2(n-q)) = 2(p+q-n-1)$ for the remaining number of one-entries. This proves that (3.57) is a lower bound, except for the case $p+q = n+1$, $p \neq q$. In this case, to achieve $Z(C^{(pq)}, B, \pi, \psi) = 2(p+q-n-1) = 0$, we would have to reduce the number of one-entries to 0. This means that in each of the $n-p$ rows and $n-q$ columns which we delete from B, we have to delete *precisely* two one-elements. Moreover, a one-element which is contained in a deleted row must not be contained in a deleted column.

Consider $b_{12} = 1$. If we delete row 1, we would delete only one one-entry; therefore b_{12} must be deleted by deleting column 2. Similarly, $b_{21} = 1$ must be deleted by deleting row 2. Elements $b_{23} = 1$ and $b_{32} = 1$ are now also deleted. Now consider $b_{34} = 1$. Row 3 contains the already deleted element b_{32} and therefore b_{34} must be deleted by deleting column 4. Similarly, we have to delete row 4, and so on. This process can only be continued until all elements are deleted if the number $n-p$ of deleted rows and the number $n-q$ of deleted columns are equal to $\frac{n-1}{2}$. Otherwise, at least one element remains undeleted, and we have $Z(C^{(pq)}, B, \phi, \psi) \geq 1$. This is precisely in accordance with (3.57). ■

Toeplitz matrices with larger bandwidth

We show that the bandwidth equal to 2 is an essential condition for Theorem 3.44. Trying to find other $0-1$ Toeplitz matrices which lead to constant permutation QAPs and have small bandwidth, the first matrices one thinks about are those of type $T^{(1)}$ and $T^{(2)}$ which are generated by the following two functions $f^{(1)}, f^{(2)}$: $\{-n+1, -n+2, \ldots, 0, \ldots, n-1\} \to \{0, 1\}$:

$$f^{(1)}(x) = \begin{cases} 1 & \text{if } x \in \{1, 2, -1, -2\} \\ 0 & \text{otherwise} \end{cases} \qquad f^{(2)}(x) = \begin{cases} 1 & \text{if } x \in \{2, -2\} \\ 0 & \text{otherwise} \end{cases}$$

The resulting matrices look as follows:

$$T^{(1)} = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 1 & 0 \end{pmatrix} \qquad T^{(2)} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 0 & 0 \end{pmatrix}$$

Both $T^{(1)}$ and $T^{(2)}$ have bandwidth 3 and are 0-1 Toeplitz matrices. We give two examples showing that $QAP(A, T^{(1)})$ and $QAP(A, T^{(2)})$ are *not* constant permutation QAPs, for $A \in$ Anti-Monge(lhG).

Example 3.7 For $n = 6$ the 6×6 matrices $T^{(1)}$, $C^{(4,4)}$, and $C^{(1,4)}$ look as follows:

$$T^{(1)} = \begin{pmatrix} 0&1&1&0&0&0 \\ 1&0&1&1&0&0 \\ 1&1&0&1&1&0 \\ 0&1&1&0&1&1 \\ 0&0&1&1&0&1 \\ 0&0&0&1&1&0 \end{pmatrix}, \quad C^{(4,4)} = \begin{pmatrix} 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&1&1&1&1 \\ 0&0&1&1&1&1 \\ 0&0&1&1&1&1 \\ 0&0&1&1&1&1 \end{pmatrix}, \quad C^{(1,4)} = \begin{pmatrix} 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 0&0&1&1&1&1 \end{pmatrix}$$

We will show that for each optimal permutation $\pi_0 \in \mathcal{S}_6$ to QAP$(C^{(4,4)}, T^{(1)})$, the set $I_{\pi_0} := \{\pi^{-1}(3), \pi^{-1}(4), \pi^{-1}(5), \pi^{-1}(6)\}$ must be equal to $\{1, 2, 5, 6\}$, whereas no optimal solution of QAP$(C^{(1,4)}, T^{(1)})$ has this property. This implies that $QAP(A, T^{(1)})$ *is not* a constant permutation QAP.

For each $\pi \in \mathcal{S}_6$, $Z(C^{(4,4)}, T^{(1)}, \pi)$ is equal to the number of the 1-entries in $T^{(1)}$ with row and column indices in the set I_π. There are $\binom{6}{2} = 15$ distinct sets I_π for all $\pi \in \mathcal{S}_6$. It can be checked that for any $I \subset \{1, 2, \ldots, 6\}$ with $|I| = 4$, the number of the 1-entries in $T^{(1)}$ with row and column indices in I is larger than or equal to 4. Equality holds only for $I = \{1, 2, 5, 6\}$. Thus, $I_{\pi_0} = \{1, 2, 5, 6\}$ holds for each optimal solution π_0 to QAP$(C^{(4,4)}, T^{(1)})$.

On the other side, for all $\pi \in \mathcal{S}_6$, $Z(C^{(1,4)}, T^{(1)}, \pi)$ equals the number of the 1-entries in $T^{(1)}$ in row $\pi^{-1}(6)$ with column indices in the set I_π. For each of the above optimal permutations π_0, $I_{\pi_0} = \{1, 2, 5, 6\}$. If we select columns $\{1, 2, 5, 6\}$ from $T^{(1)}$ we see that there exists no row with only zero entries on columns $\{1, 2, 5, 6\}$. So regardless of the value of $\pi_0^{-1}(6)$ we have $Z(C^{(1,4)}, T^{(1)}, \pi_0) \geq 1$. However, $Z(C^{(1,4)}, T^{(1)}, \psi) = 0$ for $\psi = \langle 6, 1, 2, 3, 4, 5 \rangle$, and thus π_0 is not optimal for QAP$(C^{(1,4)}, T^{(1)})$. Hence, the problems QAP$(C^{(4,4)}, T^{(1)})$ and QAP$(C^{(1,4)}, T^{(1)})$ have no common optimal solution. This completes the proof. ■

Example 3.8 For $n = 9$ the 9×9 matrices $T^{(2)}$, $C^{(4,9)}$ and $C^{(3,7)}$ look as follows:

$$T^{(2)} = \begin{pmatrix} 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \end{pmatrix} \quad C^{(4,9)} = \begin{pmatrix} 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & \cdots & 1 & 1 \end{pmatrix} \quad C^{(3,7)} = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 1 \end{pmatrix}$$

Consider the problem QAP$(C^{(4,9)}, T^{(2)})$. For any permutation $\pi \in \mathcal{S}_9$ denote $I_\pi = \{\pi^{-1}(6), \pi^{-1}(7), \pi^{-1}(8), \pi^{-1}(9)\}$. Note that $Z(C^{(4,9)}, T^{(2)}, \pi)$ is equal to the number of 1-entries in $T^{(2)}$ in the rows with indices in I_π. This observation shows that $Z(C^{(4,9)}, T^{(2)}, \pi) \geq 4$, for all $\pi \in \mathcal{S}_9$, and that the equality holds only for permutations π with $I_\pi = \{1, 2, 8, 9\}$. Thus, if π_0 is an optimal solution to QAP$(C^{(4,9)}, T^{(2)})$, then $I_{\pi_0} = \{1, 2, 8, 9\}$.

Now consider QAP$(C^{(3,7)}, T^{(2)})$. It is easy to check that a permutation ψ yields an objective function value equal to 0, $Z(C^{(3,7)}, T^{(2)}, \psi) = 0$, if and only if the equalities $\{\psi(1), \psi(5), \psi(9)\} = \{7, 8, 9\}$ and $\{\psi(3), \psi(7)\} = \{1, 2\}$ hold. Thus, for an optimal solution to QAP$(C^{(3,7)}, T^{(2)})$ these equalities must hold. Clearly, $\{\psi(1), \psi(5), \psi(9)\} = \{7, 8, 9\}$ is not fulfilled by a permutation π_0 as above, optimal solution to QAP$(C^{(4,9)}, T^{(2)})$. Thus, there exists no common optimal solution to problems QAP$(C^{(4,9)}, T^{(2)})$ and QAP$(C^{(3,7)}, T^{(2)})$. Hence, *$QAP(A, T^{(2)})$ is not* a constant permutation QAP, for $A \in$ Anti-Monge(lhG). ■

Although we do not prove it here, we remark that when n is a multiple of 4, $QAP(A, T^{(2)})$ is a constant permutation QAP, for $A \in$ Anti-Monge(lhG). The optimal permutation is obtained by "interleaving" two "copies" of π_z, one copy permuting the even indices and the other copy permuting the odd indices. More generally, for an n by n 0-1 matrix $T^{(k)}$, which has entries equal to 1 only in the k-th diagonal above and below the main diagonal, and $A \in$ Anti-Monge(lhG), $QAP(A, T^{(k)})$ is a constant permutation QAP, whenever n is a multiple of $2k$.

3.7 Applications of QAPs with an Anti-Monge and a Toeplitz matrix

In this section we describe three well known combinatorial optimization problems that can be modeled via Anti-Monge(lhG)$\times$Toeplitz(Sym): (P1) The so called "turbine problem", i.e., the assignment of given masses to the vertices of a regular polygon such that the distance of the center of gravity of the resulting system to the center of the polygon is minimized. (P2) The Traveling Salesman Problem on symmetric Monge matrices. (P3) Arrangement of data records with given access probabilities in a linear storage medium in order to minimize the average access time.

First we consider the turbine problem showing that it is $\mathcal{NP}$-hard. This implies that the general problem ANTI-MONGE(LHG)×TOEPLITZ(SYM) is $\mathcal{NP}$-hard.

Then, we investigate the two other combinatorial optimization problems and show how our polynomiality results presented by Theorem 3.30 generalizes and unifies several known results on problems (P1) and (P2).

3.7.1 The turbine problem

Hydraulic turbine runners as used in electricity generation consist of a cylinder around which a number of blades are welded at regular spacings. Due to inaccuracies in the manufacturing process, the weights of these blades differ slightly, and it is desirable to locate the blades around the cylinder in such a way that the distance between the center of mass of the blades and the axis of the cylinder is minimized. This problem was introduced by Mosevich [132] in 1986. Laporte and Mercure [112] observed that this problem can be formulated as a QAP in the following way.

The places at regular spacings on the cylinder are modeled by the vertices $v_1, \dots, v_n$ of a regular n-gon on the unit circle in the Euclidean plane, i.e., the points with coordinates

$$v_i = \Big(\sin(\frac{2i\pi}{n}), \cos(\frac{2i\pi}{n})\Big), \quad 1 \le i \le n.$$

The masses of the n blades are given by the positive reals $0 < m_1 \le m_2 \le \cdots \le m_n$. The goal is to assign the n masses to the n vertices in such a way that the center of gravity of the resulting mass system is as close to the origin as possible, i.e., to find a permutation $\phi \in S_n$ that minimizes the Euclidean norm of the vector

$$\sum_{i=1}^{n} m_{\phi(i)} \begin{pmatrix} \sin(\frac{2i\pi}{n}) \\ \cos(\frac{2i\pi}{n}) \end{pmatrix}.$$

An easy calculation reveals that minimizing the Euclidean norm of this vector is equivalent to minimizing the expression

$$\sum_{i=1}^{n}\sum_{j=1}^{n} m_{\phi(i)} m_{\phi(j)} \cos\left(\frac{2(i-j)\pi}{n}\right). \tag{3.58}$$

This is a quadratic assignment problem $QAP(A, B)$. Note that the matrix $A = (a_{ij})$ defined by $a_{ij} = m_i \cdot m_j$ is a symmetric product matrix and therefore a left-higher graded Anti-Monge matrix, since the masses m_i are sorted in nondecreasing order. The matrix $B = (b_{ij})$ defined by $b_{ij} = \cos\left(\frac{2(i-j)\pi}{n}\right)$ is a symmetric Toeplitz matrix. Note that the function $f(i) = \cos(2\pi i/n)$ which generates B is not benevolent, whereas the function $f(i) = -\cos(2\pi i/n)$ which generates $-B$ is benevolent. Therefore the original turbine problem does not fall under Theorem 3.30.

Laporte and Mercure [112] and Schlegel [167] proposed and tested several heuristics for this problem, but no fast (polynomial time) exact solution algorithm has been derived till today. We show that this is not a coincidence because in fact the problem is $\mathcal{NP}$-hard (and thus a polynomial solution algorithm would imply $\mathcal{P} = \mathcal{NP}$), as we will show in Theorem 3.47 below.

Since, on the other hand, $-B$ is benevolent, Theorem 3.30 implies that $QAP(A, -B)$ is a constant permutation QAP and π^* is its constant permutation. This corresponds to the *maximization* of (3.58). In the context of the turbine problem this means that the goal is to get the center of gravity of the mass system as far away from the origin as possible. Thus, Theorem 3.30 implies the following result.

Corollary 3.45 *The maximization version of the turbine problem, i.e., maximization of (3.58) over all permutations $\phi \in \mathcal{S}_n$, is solved to optimality by permutation π^*.* ■

We now show that the original turbine problem, i.e., the minimization of (3.58), is $\mathcal{NP}$-hard. The following simple result is needed for our proof.

Lemma 3.46 *Let $a_1, \ldots, a_{2k}$ and $b_1, \ldots, b_{2k}$ be real numbers, where $a_{2i-1}, a_{2i} < a_{2i+1}, a_{2i+2}$ holds for all $1 \le i \le k-1$, and $b_i = b_{2k+1-i}$ and $b_i > b_{i+1}$ for $1 \le i \le k$. Then the set of permutations $\pi \in \mathcal{S}_{2k}$ which minimize the expression*

$$\sum_{i=1}^{2k} a_{\pi(i)} b_i$$

contains precisely those permutations for which the equality $\{\pi(2i-1), \pi(2i)\} = \{i, 2k+1-i\}$ holds for all $1 \le i \le k$.

Proof. This statement is an extension of the well-known result of Hardy, Littlewood and Pólya stated by Proposition 1.1 and can be proved very easily, for example by an exchange argument. ■

Theorem 3.47 *The turbine problem, i.e. minimization of* (3.58) *over all permutations $\phi \in \mathcal{S}_n$, is an NP-hard problem.*

Proof. The proof is a reduction from the NP-complete Even-Odd Partition problem, (cf. Garey and Johnson [68]):

Problem: Even-Odd Partition

Instance: $2k$ positive integers $x_1, x_2, \ldots, x_{2k}$.

Question: Is there a subset $I \subseteq \{1, 2, \ldots, 2k\}$, $|I| = k$, such that the $|I \cap \{2i-1, 2i\}| = 1$ for every $i = 1, \ldots, k$, and $\sum_{i \in I} x_i = \sum_{i \notin I} x_i$?

Without loss of generality we may assume $x_{2i-1} \leq x_{2i}$ for $1 \leq i \leq k$. For the convenience of presentation, define for $1 \leq i \leq 2k+1$ the numbers

$$\alpha_i := 2i\pi/(2k+1)$$

and for $1 \leq i \leq k$ the numbers y_i by

$$y_{2i-1} = x_{2i-1}/\sin(\alpha_i) \quad \text{and} \quad y_{2i} = x_{2i}/\sin(\alpha_i).$$

Note that $0 < \alpha_i < \pi$ holds for $1 \leq i \leq k$, and thus all numbers y_i are positive, $1 \leq i \leq 2k$. Finally, let $S = \sum_{i=1}^{2k} y_i$.

Consider the following instance of the Turbine Problem. The number n of vertices on the unit circle is $2k+1$. The first $2k$ of the masses are defined by $m_{2i-1} = iS + y_{2i-1}$ and $m_{2i} = iS + y_{2i}$. Observe that by this definition

$$m_1 \leq m_2 < m_3 \leq m_4 < \cdots < m_{2i-1} \leq m_{2i} < m_{2i+1} \leq m_{2i+2} < \cdots < m_{2k-1} \leq m_{2k} \tag{3.59}$$

The value of mass $m_{2k+1} > 0$ is defined by the equation

$$m_{2k+1} + \sum_{i=1}^{k} (m_{2i-1} + m_{2i}) \cos(\alpha_i) = 0 \tag{3.60}$$

The claim is that the thereby defined instance of the turbine problem allows a mass assignment with center of gravity in the origin if and only if the instance of Even-Odd Partition has answer "Yes".

Without loss of generality assume that mass m_{2k+1} is assigned to vertex $v_{2k+1} = (1, 0)$. This induces a strong momentum towards the positive x-axis. To balance this momentum, we claim that the masses $\{m_{2i-1}, m_{2i}\}$ must be assigned to the two vertices $\{v_i, v_{2k+1-i}\}$ (in any order), for all $1 \leq i \leq k$. Let us first consider the x-coordinate of the center of gravity. It is given by the formula

$$m_{2k+1} + \sum_{i=1}^{k} (m_{\phi(i)} + m_{\phi(2k+1-i)}) \cos(\alpha_i). \tag{3.61}$$

The condition that this x-coordinate equals 0, together with (3.60), yields

$$\sum_{i=1}^{k} (m_{2i-1} + m_{2i}) \cos(\alpha_i) = \sum_{i=1}^{k} (m_{\phi(i)} + m_{\phi(2k+1-i)}) \cos(\alpha_i) \tag{3.62}$$

Note that the following relationships hold for the $\cos(\alpha_i)$, $1 \leq i \leq 2k$: $\cos(\alpha_i) = \cos(\alpha_{2n+1-i})$ for $1 \leq i \leq n$, and $\cos(\alpha_i) > \cos(\alpha_{i+1})$ for $1 \leq i \leq n-1$. Hence, the conditions of Lemma 3.46 are fulfilled. Applying this lemma we conclude that the left side of (3.62) gives the minimum of the expression on the right side over all $\phi \in \mathcal{S}_{2n}$, and moreover, the right side is equal to this minimum if and only if $\{\pi(2i-1), \pi(2i)\} = \{i, 2k+1-i\}$, for all $1 \le i \le k$, i.e., the masses $\{m_{2i-1}, m_{2i},\}$ are assigned to the vertices $\{v_i, v_{2k+1-i}\}$.

Secondly, let us consider the y-coordinate of the center of gravity. The above argument implies that in the corresponding formula the coefficient of masses m_{2i-1}, m_{2i} are either $\sin(\alpha_i)$, $-\sin(\alpha_i)$ or $-\sin(\alpha_i)$, $\sin(\alpha_i)$, respectively. Hence the total value contributed to the y-coordinate of the center of gravity by these two masses is either

$$\sin(\alpha_i)m_{2i-1} - \sin(\alpha_i)m_{2i} = x_{2i-1} - x_{2i}$$

or

$$-\sin(\alpha_i)m_{2i-1} + \sin(\alpha_i)m_{2i} = -x_{2i-1} + x_{2i}.$$

With this it is easy to see that the y-coordinate can be zero if and only if there is a solution of Even-Odd Partition. The set I contains those indices whose corresponding masses are assigned to vertices $v_1, \dots, v_k$.

The above arguments assumed exact calculations with real numbers. To make the proof valid, one has to work with sufficiently precise rational approximations of sines and cosines. The condition in the claim must be modified: Instead of insisting that the center of gravity of a mass assignment lies exactly in the origin, we have to require that its distance from the origin is smaller than some given threshold ε. Since the values of the right-hand side of (3.62) which are not equal to the minimum can be bounded away from zero, it is possible to work out such a threshold ε and the precision requirement for the computations in polynomial time. We omit the details.
■

3.7.2 Two further applications

This section deals with the Traveling Salesman Problem (P2) on symmetric Monge distance matrices, and with a data arrangement problem (P3). Theorem 3.30 can be applied to both problems and yields short proofs for known results on problems (P2) and (P3).

The TSP on symmetric Monge matrices

The *Traveling Salesman problem* (TSP) consists in finding a shortest closed tour through a set of cities with given distance matrix. This problem is a fundamental problem in combinatorial optimization and well-known to be NP-hard. For more information, the reader is referred to the comprehensive book edited by Lawler, Lenstra, Rinnooy Kan, and Shmoys [116]. Several special cases of the TSP are known to be solvable in polynomial time due to special combinatorial structures in the distance matrix. The following proposition states one of the first results on easy special cases of the TSP which was obtained by Supnick [172]. We show that this result can be easily derived as a corollary of Theorem 3.30.

Proposition 3.48 *(Supnick [172], 1957)*
Every instance of the TSP with a symmetric Monge distance matrix $D = (d_{ij})$ is solved by the permutation π^*.

Proof Let $\Delta = 2\max_{1\le i,j\le n}\{|d_{ij}|\}$ and define a sum matrix $S = (s_{ij})$ by $s_{ij} = (i+j)\Delta$. Moreover, let us define a symmetric Toeplitz matrix B by its generating benevolent function $f(1) = f(-1) = f(n-1) = f(-n+1) = -1$ and $f(i) = 0$ for $i \notin \{-n+1, -1, 1, n-1\}$. The proof relies on the following three simple steps.

Firstly, QAP$(S-D, B)$ is solved by π^*: Since S and $-D$ both are Anti-Monge matrices, so is $S-D$. Moreover, it is straightforward to verify that the matrix $S-D = (c_{ij})$ is left-higher graded. Since B is a symmetric Toeplitz matrix generated by a benevolent function f, Theorem 3.30 applies.

Secondly, QAP$(-S, B)$ is solved by π^*: Since $-S$ is a sum matrix and B is a circulant, Theorem 3.11 implies that QAP$(-S, D)$ is solved by every permutation $\pi \in S_n$.

Finally, based on Observation 3.1, we can add $S-D$ and S and get that QAP$(-D, B)$ is solved by π^*. Since QAP$(-D, B)$ and QAP$(D, -B)$ are equivalent, the problem QAP$(D, -B)$ is also solved by π^*.

Now, it is easily checked that matrix $-B$ is the adjacency matrix of an undirected cycle on n vertices. Hence, QAP$(D, -B)$ exactly corresponds to the TSP with distance matrix D. ■

Data arrangement in a linear storage medium

Consider a set of n records $r_1, \ldots, r_n$ which are referenced repetitively, where with probability p_i the reference is to record r_i and where different references are independent. Without loss of generality the records are numbered such that $p_1 \le p_2 \le \cdots \le p_n$. The goal is to place these records into a linear array of storage cells, like a magnetic tape, such that the expected distance between two consecutively referenced records is minimized, i.e., one wishes to minimize

$$\sum_{i=1}^{n}\sum_{j=1}^{n} p_{\pi(i)}p_{\pi(j)}d_{ij} , \tag{3.63}$$

where d_{ij} is the distance between the records placed at storage cells i and j respectively. In the late 1960s and early 1970s, much research has been done on the special case of this problem where the distance d_{ij} is given as $d_{ij} = f(|i-j|)$, $f\colon \{0, \ldots, n-1\} \to \mathbb{R}$, i.e., d_{ij} only depends on the absolute value of the difference between i and j. The following proposition summarizes three of these results in order of increasing generality.

Proposition 3.49 *(a) If $d_{ij} = |i-j|$, then the data arrangement problem is solved by permutation π^*.* (Timofeev and Litvinov [178], 1969)

(b) *If $d_{ij} = f(|i-j|)$ with nondecreasing and convex function f, then the data arrangement problem is solved by permutation π^*.* (Burkov, Rubinstein and Sokolov [35], 1969)

(c) *If $d_{ij} = f(|i-j|)$ with nondecreasing function f, then the data arrangement problem is solved by permutation π^*.* (Metelski [127], Pratt [145], 1972) ■

Metelski [127] and Pratt [145] realized that the above results are all contained in the following result due to Hardy, Littlewood and Pólya, here formulated in the language of the QAP and proved by applying our Theorem 3.30.

Proposition 3.50 (Hardy, Littlewood and Pólya [86], 1926)
Let $A = (a_{ij})$ be defined by $a_{ij} = x_i y_j$ for nonnegative real numbers $x_1 \leq \cdots \leq x_n$ and $y_1 \leq \cdots \leq y_n$. Let $B = (b_{ij})$ be a symmetric Toeplitz matrix generated by a function f that is nondecreasing on $\{0, \ldots, n\}$. Then QAP(A, B) is solved by π^*.

Proof. It is easy to verify that matrix A is a monotone Anti-Monge matrix. Since moreover matrix B is generated by a benevolent function f, Theorem 3.30 can be applied. ■

Motivated from this data arrangement problem, Rubinstein [159] showed that the result stated by Proposition 3.50 holds also for left-higher graded Anti-Monge matrices instead of product matrices. Obviously, this result is also contained as a special case in Theorem 3.30.

3.8 Open problems and conclusions

In this chapter we investigated restricted versions of the QAP where the coefficient matrices possess special combinatorial properties. We tried to clearly separate polynomially solvable cases from $\mathcal{NP}$-hard ones. In this context we first considered QAPs where both coefficient matrices belong either to the matrix class Monge or the matrix class Anti-Monge. It is shown that this restriction does not simplify the problem in general, i.e., QAP(A,B) where both A and B are Monge (or Anti-Monge) matrices remains $\mathcal{NP}$-hard. Generally, even coefficient matrices which also possess additional properties do not yield polynomially solvable cases of QAP, as for example NegProduct(Sym)×NegProduct(Sym) and NegProduct(Sym)×NegChess.

Only in the "very special" case where both matrices $-A$ and $-B$ are chessboard matrices, the problem QAP(A,B) is polynomially solvable.
The scenario of QAPs with a Monge matrix and an Anti-Monge matrix is more optimistic. The problem Product(Sym)×NegProduct(Sym) and the even-sized instances of Chess×Monge are polynomially solvable, whereas the complexity of the odd sized instances of the latter problem remains an open question. Here, an

interesting open question concerns the computational complexity of the more general problem Monge×Anti-Monge. Proving its polynomiality would complete Table 3.1.

Further, we have shown that the QAP with a circulant and a Monge matrix remains $\mathcal{NP}$-hard, although there is at least one solvable case of it, namely the TSP on Monge matrices. We singled out several solvable cases of QAPs with one Monge matrix and the other a small bandwidth matrix. Solvable cases arising from the taxonomy problem are also classified in this group of problems. It is worthy to notice that in all of these solvable cases the Monge matrix fulfills also additional properties and all of them are constant permutation QAPs. The big open question in this group of QAPs is the complexity of the problem Circulant×Circulant. Recall that Circulant×Circulant contains as special case the so called *circulant TSP*.

Finally, we investigated in detail the QAP with a left-higher graded Anti-Monge matrix and a Toeplitz matrix. We show that this problem is $\mathcal{NP}$-hard and identify three conditions on the Toeplitz matrix which yield, in turn, constant permutation QAPs. Among these solvable cases, the most celebrated arise when the Toeplitz matrix is generated by a benevolent function or a k-benevolent function. In the other solvable case the Toeplitz matrix has bandwidth two, whereas for Toeplitz matrices with larger bandwidth the problem Anti-Monge(lhG)×Toeplitz is shown to be $\mathcal{NP}$-hard.
Considering the problem Anti-Monge(lhG)×Toeplitz, we believe that much more can be done to derive other properties of the Toeplitz matrix which yield constant permutation QAPs. The results presented here are only the first steps in this direction. However, deriving a characterization of all Toeplitz matrices which yield constant permutation problems from the class Anti-Monge(lhG)×Toeplitz is an open problem whose complete solution is currently out of sight.
Considering the applications of the QAP with a left-higher graded Anti-Monge and a Toeplitz matrix, we notice that the results on this restricted version of the QAP generalize and unify some well known results on the TSP and on a practical data arrangement problem. Moreover, the QAP formulation of another well known problem, namely the turbine problem, belongs to the class Anti-Monge(lhG)×Toeplitz. We show that the turbine problem is $\mathcal{NP}$-hard, answering in this way a question which has been open for a long time.

Summarizing, the results presented in this chapter bear evidence of the fact that the QAP is a very difficult problem also from a theoretical point of view. Much structure on the coefficient matrices is required in order to obtain polynomially solvable cases. Some of the solvable cases are inherently linear assignment problems (eg. Product(Sym)×Product(Sym)), whereas others of them are constant QAPs and thus, somehow inherently trivial. The most interesting solvable cases are *constant permutation QAPs* which can be solved in $O(n)$ time, where n is the size of the problem. There are only a few solvable cases which cannot be classified to any of these groups. In our opinion, the fact that almost all solvable cases of the problem discussed in this chapter are either constant QAPs or constant permutation QAPs is very intriguing. Is this behavior due to our poor knowledge on the problem or is this an inherent characteristic of it? Unfortunately, there is no hope to answer this question without identifying other solvable and provably hard cases of the problem at hand.

The method used to single out the constant permutation of constant permutation QAPs and hence, to prove their polynomiality, can be specified as *reduction into extremal rays*. In order to use this method eventual hints are considered that suggest an eventual constant permutation for the version of the problem at hand. In turn, we make use of the advantageous combinatorial structure of the problem data. Namely, exploiting the fact that the corresponding matrix classes form cones, the investigations can be restricted on simpler instances of the problem with 0-1 coefficients only. Proving that the guessed constant permutation is really an optimal solution to each of these problem instances is done, in most of the cases, by a simple but peculiar case-analysis or by exchange arguments. Often, the relaxation RQAP of the problem has been considered, where the rows and the columns of the corresponding coefficient matrix are permuted independently by two different permutations, say π and ψ. Then, it is shown that the RQAP is also a constant permutation problem with an optimal solution which permutes both rows and columns by the same permutation, say π_0. Clearly, the permutation π_0 is then an optimal solution to the original QAP. The constant QAPs or QAPs which inherently are linear assignment problems are usually quite easy to handle. The work to be done amounts on recognizing these properties by finding an appropriately equivalent reformulation of the problem. While analyzing the methods used in this chapter, we should notice that in one single case, namely when dealing with problems of the class Large(Sym)$\times$NegChess, a dynamical programming approach is applied.

We conclude this chapter by pointing out that we have only thrown a few beams of light on the shady border area between easy and hard cases of QAPs, drawing just a dashed tiny part of the desired borderline.

Chapter 4

QAPs Arising from Optimization Problems in Graphs

In this chapter we consider some optimization problems in graphs which can be formulated as QAPs. Several polynomially solvable special cases have been identified for these problems which are generally $\mathcal{NP}$-hard. We present a number of well known results of this type translated in the language of QAPs. We consider also QAPs which arise from *graph packing*. The proofs of some theoretical results on graph packing problems lead to algorithmic approaches for solving the corresponding special cases of the QAP. Further on, we present some $\mathcal{NP}$-hard and some polynomially solvable cases of *the general quadratic assignment problem* GQAP introduced in Subsection 1.3.6. Most of the special cases of GQAP considered here occur in the frame of an optimization problem in isomorphic graphs, namely the problem of finding an isomorphism with "minimum weight" for two given isomorphic graphs. Finally, we formulate several open questions which arise when dealing with the special versions of the QAP introduced along this chapter.

The chapter is organized as follows. In the first section we introduce so called *placement problems*, concentrating on restricted versions of the *linear arrangement problem*. In this context, some well known results which lead to solvable cases of the QAP are reviewed. In the second section we consider special cases of the QAP arising from the *feedback arc set problem*. It follows a detailed analysis of QAP formulations of graph packing problems and their complexity analysis in Section 4.3. Finally, in the fourth section some results on polynomially solvable and provably hard cases of the GQAP are summarized. We conclude by proving that the so called *pyramidal QAP* is $\mathcal{NP}$-hard, in contrast to the *pyramidal TSP* which is known to be polynomially solvable (see for example [116]).

4.1 Special cases related to placement problems

In *placement problems* we are given n electronic modules that have to be placed in a one dimensional array at unit intervals. Every two modules are connected by a number of wires. Let a_{ij} denote the number of wires connecting the modules i and j. The objective is to place the modules in such a way that the overall length of the connecting wires is minimized. Clearly, this problem belongs to Matrix×Toeplitz(Sym). The $n \times n$ matrix $A = (a_{ij})$ is symmetric and $B = (b_{ij})$ is a symmetric Toeplitz matrix defined by $b_{ij} = |i-j|$, $1 \le i,j \le n$. This problem was originally introduced in 1972 by Hanan and Kurtzberg [82] under the name "the module placement problem" or "backboard wiring problem". The special case of the above problem, where matrix A is the adjacency matrix of a graph is known as *linear ordering* or *linear arrangement* problem (this problem is also known as *minimum sum labeling* problem). In general, this problem is $\mathcal{NP}$-complete (see [68]). Evan and Shiloach [56] have shown that even the restrictions of the linear arrangement problem in acyclic digraphs or in bipartite graphs are $\mathcal{NP}$-complete. Consider a topological labeling of an acyclic digraph G on n vertices (i.e., the tail of every arc has smaller label than its head). If $A = (a_{ij})$ is the adjacency matrix of G than $a_{ij} = 0$ for all $i \le j$, $1 \le i,j \le n$. Now, the result of Evan and Shiloach formulated in the QAP language, looks as follows:

Proposition 4.1 *(Evan and Shiloach [56], 1975)*
Let the $n \times n$ matrix $B = (b_{ij})$ be given by $b_{ij} = |i-j|$, $1 \le i,j \le n$. Consider an $n \times n$ matrix $A = (a_{ij})$ with 0-1 entries fulfilling the condition $a_{ij} = 0$ for $i \le j$, $1 \le i,j \le n$. Then QAP(A,B) is $\mathcal{NP}$-complete. ∎

Through the rest of this section we summarize several polynomiality results on the linear arrangement problem formulated in the language of QAP. The first polynomial result on this problem concerns (undirected) trees. Namely, in 1976 Goldberg and Klipker [77] proved that the linear arrangement problem on (undirected) trees is polynomially solvable. They proposed an $O(n^3)$ algorithm, where n is the size of the problem, i.e., the number of vertices of the tree. In 1979 Shiloach [168] derived a new algorithm improving the time complexity to $O(n^{2.2})$. Finally, in 1984 Chung [43] improved the time complexity to $O(n^\lambda)$, where λ can be chosen to be any real which satisfies $\lambda > \log 3/\log 2 \approx 1.6$.

Proposition 4.2 (Goldberg and Klipker [77], 1976, Shiloach [168], 1979, Chung [43], 1984)
Consider the $n \times n$ matrices A and $B = (b_{ij})$, where A is the adjacency matrix of a tree on n vertices and $b_{ij} = |i-j|$, $1 \le i,j \le n$. QAP(A,B) is solvable in polynomial time, namely, its solution involves $O(n^\lambda)$ elementary operations, where λ is as described above. ∎

Both Chung and Shiloach propose recursive algorithms where the optimal linear arrangement for the given tree results as an appropriate combination of optimal linear arrangements for some appropriately chosen rooted subtrees. The improvement in the time complexity results from a more efficient use of the subtrees in the recursive process rather then from some principally new idea. Note that the result of Proposition 4.2 applies only in the case that matrix $A = (a_{ij})$ is a 0-1 matrix, i.e., the entries a_{ij} are either equal 0 or equal to 1. None of the above algorithms works in the case that A is the *weighted* adjacency matrix of a weighted tree. To the best of our knowledge the time complexity of $QAP(A, B)$ in this case remains an open question.

A polynomially solvable version of the linear arrangement problem, where A is the weighted adjacency matrix of a *rooted* tree, was identified by Adolphson and Hu [1]. In this version of the problem not all permutations are considered as feasible arrangements. A feasible arrangement π has the property that for each arc (i, j) in the rooted tree, vertex i is be embedded to the left of vertex j, i.e., $\pi(i) < \pi(j)$. Note, that because of this additional restriction this problem is a GQAP. Adolphson and Hu [1] propose a rather technical $O(n \log n)$ algorithm for this problem.

Proposition 4.3 (Adolphson and Hu [1], 1973)
Let $A = (a_{ij})$ be the $n \times n$ weighted adjacency matrix of a rooted tree, where the entries which correspond to non-existing edges are equal to 0. Let the $n \times n$ matrix $B = (b_{ij})$ be given by $b_{ij} = |i - j|$, $1 \leq i, j \leq n$. Denote by $\mathcal{S}_n^A$ the set of those permutations π of $\{1, 2, \ldots, n\}$ such that $\pi(i) < \pi(j)$ for each pair of indices i, j with $a_{ij} \neq 0$. Then, the problem $GQAP(A, B, \mathcal{S}_n^A)$ given below is solvable in $O(n \log n)$ time.

$$\min_{\pi \in \mathcal{S}_n^A} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij} \qquad \blacksquare$$

Another polynomially solvable version of the linear arrangement problem occurs in the context of network flows. Assume that $A = (a_{ij})$ is the weighted adjacency matrix of a symmetric, undirected, edge weighted network. Gomory and Hu [78] defined the so-called Gomory-Hu tree that gives a concise representation of the maximum flows between any pair of vertices in the network. For any two vertices i and j in the network, the value of the maximum flow from i to j equals the weight of the shortest edge on the path joining i to j in the Gomory-Hu tree. Actually, the weights on the edges of the network are the capacities involved in the related max-flow problems.

Proposition 4.4 (Adolphson and Hu [1], 1973)
If the Gomory-Hu tree associated with an $n \times n$ matrix A is a path and matrix $B = (b_{ij})$ is defined by $b_{ij} = |i - j|$, $1 \leq i, j \leq n$, then $QAP(A, B)$ is polynomially solvable.

Proof (Sketch).
This result is derived by Adolphson and Hu [1] as a corollary from a more general result. Let N_A be a symmetric, undirected, (edge) weighted network with weighted

adjacency matrix A. It is proven that the overall sum of the edge weights of the Gomory-Hu tree corresponding to N_A is a lower bound on the objective function value of $QAP(A,B)$. In the case that the Gomory-Hu tree is a path, it is shown that the sum of its edge weights equals $Z(A,B,\pi)$, where $\pi(i)$ denotes the position of vertex i on the Gomory-Hu path. Thus, such a permutation π is an optimal solution to $QAP(A,B)$. (Clearly, there exist at least two optimal solutions π_1 and π_2 which are related by the equality $\pi_1(i) = \pi_2(n-i+1)$, for all $i = 1,2,\dots,n$.) Thus, in this case, solving $QAP(A,B)$ amounts to computing the Gomory-Hu path of for the network N_A. As the Gomory-Hu tree for a given symmetric, undirected, edge weighted network can be computed in polynomial time ($O(n^3)$, e.g. see [78]), $QAP(A,B)$ is polynomially solvable. ■

It would be nice to have a characterization of networks whose Gomory-Hu tree is a path, or at least sufficient or necessary conditions for a symmetric, edge weighted, undirected network to have this property. To the best of our knowledge this interesting problem, which does not seem to be easy, has not been investigated up to now.

To conclude this section notice that in the case that A is an Anti-Monge left-higher graded matrix, the corresponding placement problem is solved by the permutation π^* defined in the the previous chapter. This is a straightforward corollary of Theorem 3.30, since matrix $B = (|i-j|)$ is a symmetric Toeplitz matrix generated by a benevolent function.

4.2 Special cases related to the feedback arc set problem

In the *minimum weight feedback arc set* problem, FAS, a weighted digraph $G = (V,E)$ with vertex set V and edge set E is given. The goal is to remove a set of arcs with minimum overall weight from E such that all directed cycles (dicycles) in G are destroyed and an acyclic subgraph remains. Clearly, the minimum weight feedback arc set problem is equivalent to finding an acyclic subgraph of G with maximum weight. It is worthy to notice here that this problem was sometimes termed as *linear ordering problem*, as for example in [148]. Notice however that the problem considered in this section is completely different from that treated in the previous section under the name *linear ordering (arrangement) problem.* We have adopted the terminology used (among others) by Garey and Johnson in [68] for both the linear ordering problem and the minimum weight feedback arc set problem. The unweighted version of FAS, (that is, with weights equal to 0 or 1) is called also *the acyclic subdigraph problem* [94]. An interesting application of FAS is the so called *triangulation of the input-output tables* which arises along with input-output analysis in economics (details and further references can be found in [148]).

The minimum weight feedback arc set problem can be formulated as a QAP. Since the vertices of an acyclic subgraph can be labeled topologically (i.e. the tail of every arc a has smaller label than its head), the FAS is equivalent to $QAP(A,B)$ where matrix A is the weighted adjacency matrix of G and matrix B equals the *lower triangular matrix* $L_n = (\ell_{ij})$, where $\ell_{ij} = 1$ if $i \geq j$ and $\ell_{ij} = 0$, otherwise.

The problem FAS is well known to be $\mathcal{NP}$-complete (see Karp [100], Garey and Johnson [68]). Gavril [70] proved $\mathcal{NP}$-completeness even for the case of an unweighted digraph where every vertex has in-degree and out-degree at most three.

Proposition 4.5 (Gavril [70], 1977)
Let A be an $n \times n$ zero-one matrix where every row and every column contains at most three entries of value one. Then, the problem $QAP(A, L_n)$ is $\mathcal{NP}$-complete. ■

Through the rest of this section we summarize some results on polynomially solvable cases of the problem $QAP(A, L_n)$. The basic stone for most of the polynomiality results on FAS presented in this section is a fundamental minimax theorem of Lucchesi and Younger [123], which probably is the very first result in this direction. Most of the later results on this problem, both algorithmic and theoretical ones, make use of this theorem or generalize its basic idea. Before formulating this result of Lucchesi and Younger, we need the definition of a *dicut* and the definition of a *transversal (of dicuts)*.

Definition 4.1 **(a)** *Consider a digraph $G = (V, E)$, with vertex set V and edge set E. A* directed cut (dicut) *in G is a set of arcs with tail in S and head in $V \setminus S$, provided that there are no arcs going from $V \setminus S$ to S and S is a strict subset of V, $S \subset V$, $S \neq \emptyset$, $S \neq V$.*

(b) *Consider a set of subsets of E, $\{E_1, E_2, \ldots, E_k\}$, $E_i \subseteq E$, $1 \leq i \leq k$, $k \in \mathbb{N}$. A* transversal of $\{E_1, E_2, \ldots, E_k\}$ *in G is a set of arcs E', $E' \subseteq E$, with the property that each of the sets E_i, $1 \leq i \leq k$, contains at least one arc from E'. In the case that $\{E_1, E_2, \ldots, E_k\}$ is the set of $\underline{\text{all}}$ dicycles or $\underline{\text{all}}$ dicuts in G, we have a transversal of dicycles or a transversal of dicuts, respectively.*

(c) *Assume that G is a weighted digraph with a weight function $w\colon E \to \mathbb{N}$, $e \mapsto w(e)$. The weight of a set of arcs E', $E' \subseteq E$, is given as sum of the weights of all its elements: $w(E') = \sum_{e \in E'} w(e)$.*

Proposition 4.6 (Lucchesi [122], 1976, Lucchesi and Younger [123], 1978)
Consider a finite digraph G. The minimum cardinality of a transversal of dicuts in G equals the maximum cardinality of a collection of pairwise disjoint dicuts. ■

Example 4.1 An illustrating example for dicuts, transversals and for Proposition 4.6.
Consider the digraph presented in Figure 4.1. It is easy to see that there are only three dicuts in this digraph: $D_1 = \{(1,2),(1,3)\}$, $D_2 = \{(1,3),(2,3),(2,4)\}$ and

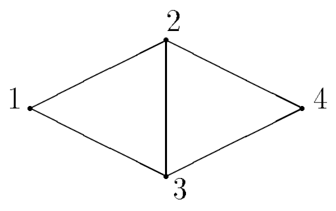


Figure 4.1: An illustrative example

$D_3 = \{(2,4),(3,4)\}$. The set of arcs $T = \{(1,2),(2,3),(2,4)\}$ is a transversal of these dicuts, since $T \cap D_i \neq \emptyset$, for $i = 1,2,3$. Notice that $D_1 \cap D_2 \cap D_3 = \emptyset$ and $D_2 \cap D_3 = \{(2,4)\}$. Thus, the maximum cardinality of a collection of pairwise disjoint dicuts equals 2. According to Proposition 4.6 there exists a transversal of dicuts with cardinality equal to 2. Indeed, the set $\{(1,3),(2,4)\}$ is such a transversal. According to Proposition 4.6 there exist no transversal with cardinality 1. Indeed, the existence of such a transversal would imply $D_1 \cap D_2 \cap D_3 \neq \emptyset$. ■

Later on, several alternative, sometimes constructive proofs for Proposition 4.6 have been given, eg. [61, 101, 121]. Moreover, this basic theorem can be generalized for weighted digraphs as shown in [50, 61, 122].

Proposition 4.7 (Lucchesi [122], 1976, Edmonds and Giles [50], 1977, Frank [61], 1981)
Consider a finite weighted digraph $G = (V,E)$ with nonnegative integral weights $w(e)$ on its edges $e \in E$. The minimum weight of a transversal of dicuts in G equals the maximum weight of a collection $\mathcal{C}$ of directed cuts with the property that e occurs in at most $w(e)$ dicuts from $\mathcal{C}$, for all $e \in E$. ■

The following proposition, which was implicitly proven by many authors, eg. [61, 65, 122] is a corollary of Proposition 4.7.

Proposition 4.8 (Lucchesi [122], 1976, Frank [61], 1981, Gabow [65], 1993)
For the weighted adjacency matrix A of a weighted planar digraph $G = (V,E)$, $QAP(A, L_n)$ is solvable in polynomial time.

Proof. Let $G = (V,E)$ be a weighted planar digraph with weights $w(e)$ on its edges $e \in E$ and let G^* be its dual digraph. To each arc e' of the dual digraph G^* join a weight $w(e')$ equal to the weight of its corresponding arc e in G, $w(e') = w(e)$. It is well known that the one to one correspondence between arcs of G and arcs of G^* implies a one to one correspondence between directed cycles (dicycles) in G and dicuts in G^*. Clearly, there is also a one to one correspondence between transversals of dicycles in G and transversals of dicuts in G^*. Under these conditions, corresponding objects (dicuts and dicycles, transversals of dicuts and transversals of dicycles) have the same weight. Thus, as a corollary of Proposition 4.7, we get:

The minimum weight of a transversal of dicycles in a weighted planar digraph $G = (V, E)$, with nonnegative integral weights $w(e)$ on its edges $e \in E$, equals the maximum weight of a collection $\mathcal{C}$ of dicycles in G, with the property that e occurs in at most $w(e)$ dicycles from $\mathcal{C}$, for all $e \in E$.
Solving $QAP(A, L_n)$ is equivalent to finding a transversal of dicycles in G with minimum weight. Indeed, once such a transversal of dicycles is found, the arcs of this transversal are canceled from G and the remaining graph contains no dicycles. Hence, this graph can be sorted topologically (in polynomial time) and the corresponding labeling of vertices yields an optimal solution to $QAP(A, L_n)$. On the other side, finding a transversal of dicycles in G with minimum weight is equivalent to finding a transversal of dicuts in G^* with minimum weight. Lovasz [121] has pointed out that finding a minimum weight transversal of dicuts in G^* is equivalent to finding a minimum weight set of arcs whose contraction transforms G^* in a strongly connected digraph. Finally, Frank [61] derived an $O(n^5)$ combinatorial algorithm for finding such a set of arcs in an arbitrary weighted digraph on n vertices. ■.

As far as we know, the best algorithm for solving $QAP(A, L_n)$, where A is the weighted adjacency matrix of a planar digraph, is due to Gabow [65]. Its time complexity is $O(n^3)$, where n is the size of the considered QAP.
Notice that the problem $QAP(A, L_n)$ of size n where A is a symmetric matrix is trivial: in this case $QAP(A, L_n)$ is a constant QAP. This version of the QAP corresponds to the weighted feedback edge set problem for (undirected) graphs with a symmetric weighted adjacency matrix.

In Proposition 4.8 the condition that A is a weighted adjacency matrix of a planar digraph is a sufficient condition for the polynomial solvability of $QAP(A, L_n)$, but not a necessary one. The remaining polynomiality results to be presented in this section confirm this fact. The propositions formulated here below are results on FAS or on its equivalent, the so called *acyclic subdigraph problem*, translated in the language of QAP.

Proposition 4.9 (Ramachandran [147], 1988)
Let A be the weighted adjacency matrix of a reducible flow graph on n vertices. Then, $QAP(A, L_n)$ is solvable in $O(n^2 m \log(\frac{n^2}{m}))$ steps, where m is the number of the nonzero entries in matrix A. In the case that A is the $0-1$ adjacency matrix of a reducible flow graph, $QAP(A, L_n)$ can be solved in $O(m^2)$ time, where m is the same as above. ■

Proposition 4.10 (Penn and Nutov [137], 1994)
Let A be the weighted adjacency matrix of a $K_{3,3}$-free digraph on n vertices. Then, $QAP(A, L_n)$ is solvable in $O(n^3)$ time. ■

Proposition 4.10 can also be derived as a corollary of a stronger result of Grötschel, Jünger and Reinelt [81]. To formulate this stronger result we need one recent notion from the graph theory, namely the notion of a *weakly acyclic digraph* (cf. [81]). As the definition is not straightforward (it makes use of the acyclic subgraph polytope) we omit it here. We only remark that planar digraphs and $K_{3,3}$-free digraphs are weakly acyclic graphs (see [81] and [9], respectively). For more details on this topic and for an exact definition of *weakly acyclic digraphs* the reader is referred to [9, 81].

Proposition 4.11 (Grötschel, Jünger and Reinelt [81], 1985)
If A is the weighted adjacency matrix of a weakly acyclic digraph on n vertices, then $QAP(A, L_n)$ is solvable in polynomial time. ■

Remarks. 1. The algorithm proposed by Grötschel et al. [81] uses the ellipsoid method, whereas the algorithm derived by Penn and Nutov in [137] is a combinatorial one.
2. What about the recognition of the well solvable cases of QAP described above? Obviously the polynomial recognition algorithms for planar graphs, $K_{3,3}$-free digraphs and reducible flow graphs (see for example [176, 66]) can be used for recognizing the corresponding properties of the coefficient matrix A for a given $QAP(A, B)$. To the best of our knowledge, there exists no polynomial time algorithm for the recognition of weakly acyclic digraphs and the computational complexity of this problem is an open question.

We conclude this section by a straightforward corollary of Theorem 3.8:

Corollary 4.12 *For a right-higher graded matrix A the problem $QAP(A, L_n)$ is polynomially solvable.*

Proof. The claim follows from Theorem 3.8 when considering that L_n is a left-lower graded matrix. ■

Concluding this section we single out a probably fruitful direction of further research on QAPs related to FAS. Namely, polynomial solvable cases of $QAP(A, L_n)$ may probably arise in the case that matrix A has some "nice" combinatorial properties. By exploiting special combinatorial structures of matrix A fast algorithms for special cases of $QAP(A, L_n)$ can then be derived. For example, the problems Monge$\times L_n$, Anti-Monge$\times L_n$ and their complexity might be the next "simple" questions to investigate.

4.3 QAPs arising from graph packing problems

Another line of research on well solvable cases of QAP may be started from the theory of *graph packing*, cf. Bollobás [18]. Let us first introduce the graph packing problem. Consider two weighted (undirected) graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ with vertex sets $V_1 = \left\{v_i^{(1)} \mid 1 \le i \le n\right\}$ and $V_2 = \left\{v_i^{(2)} \mid 1 \le i \le n\right\}$ and edge sets E_1 and E_2, respectively. Denote by $A = (a_{ij})$ and $B = (b_{ij})$ the weighted adjacency matrices of G_1 and G_2, respectively. Throughout this section, we assume that the weights on the edges of both graphs are positive. Moreover, we set $a_{ij} := 0$ or $b_{ij} := 0$ in the case that $\left(v_i^{(1)}, v_j^{(1)}\right) \notin E_1$ or $\left(v_i^{(2)}, v_j^{(2)}\right) \notin E_2$), respectively.

Definition 4.2 *A permutation $\pi \in \mathcal{S}_n$ is called a* packing of G_2 into G_1 *if* $\left(v_i^{(2)}, v_j^{(2)}\right) \in E_2$ *implies* $\left(v_{\pi(i)}^{(1)}, v_{\pi(j)}^{(1)}\right) \notin E_1$*, for* $1 \le i, j \le n$.

In other words, a packing of G_2 into G_1 is an embedding of the vertices of G_2 into the vertices of G_1 such that no pair of edges coincide. For two graphs G_1 and G_2 as above the *graph packing problem* consists of finding a packing of G_2 into G_1, if one exists, or proving that no packing exists, otherwise.

Example 4.2 Packing and edge coincidences.
Consider a connected graph $G = (V, E)$ and its complement $\bar{G} = (V, \bar{E})$, where V is the vertex set of both G and $\bar{G}$, $V = \{v_1, v_2, \ldots, v_n\}$, and $\bar{E} = \{(v_i, v_j) \colon 1 \le i, j, \le n\ ,\ (v_i, v_j) \notin E\}$. Clearly, the identity permutation is a packing of $\bar{G}$ into G. In the case that G is complete each embedding of the vertices of $\bar{G}$ into the vertices of G is a packing. Otherwise, let v_i be a vertex with degree smaller than $n-1$ in G. Denote by v_j and v_k two vertices such that $(v_i, v_j) \in E$ and $(v_i, v_k) \notin E$. A permutation π such that $\pi(j) = k$, $\pi(k) = j$ and $\pi(t) = t$ for all $t \notin \{j, k\}$ is not a packing of $\bar{G}$ to G. Indeed, $(v_i, v_k) \in \bar{E}$ and $(v_{\pi(i)}, v_{\pi(k)}) \in E$. In this case we have an edge coincidence. ■

Clearly, the graph packing problem is related to the QAP. Consider an arbitrary embedding π of G_2 into G_1, that is a one to one mapping of the vertices of G_1 into the vertices of G_2. Assume that there exists a pair of indices (i, j) such that $\left(v_i^{(2)}, v_j^{(2)}\right) \in E_2$ and $\left(v_{\pi(i)}^{(1)}, v_{\pi(j)}^{(1)}\right) \in E_1$. We will refer to this fact saying that an *edge coincidence* occurred. For an embedding π, the number of pairwise different pairs (i, j) which yield an edge coincidence as above is called *number of edge coincidences*. Obviously the objective function value of $QAP(A, B)$ corresponding to a packing π is equal to zero:

$$Z(A, B, \pi) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij} = 0$$

Conversely, it is clear that a permutation $\pi \in \mathcal{S}_n$ such that $Z(A, B, \pi) = 0$ is a packing of G_2 into G_1. Thus, solving the graph packing problem for two given graphs G_1 and G_2, with weighted adjacency matrices A and B, respectively, is equivalent to finding the optimal value of $QAP(A, B)$ and an optimal solution of it in the case that this optimal value equals 0.

This section is a summary of results on graph packing translated in the language of QAPs. Most of them concern polynomially solvable cases of the QAP. Let $\Delta(G)$ denote the maximal degree of vertices of a graph G through the rest of this section.

The following theorem concerns "sparse" QAPs, i.e., QAPs whose coefficient matrices have together a "large number" of zero entries. It is intuitively clear that if the overall number of zero entries of matrices A and B is very large, then the optimal value of $QAP(A,B)$ is 0. Also by intuition, it should be "easy" to find an optimal solution to such a $QAP(A,B)$. The following theorem confirms the intuition and tries to mathematically precise what "large number" of zero entries means.

Theorem 4.13 (Bollobás and Eldrige [19], 1978)
Let A and B be two symmetric $n \times n$ matrices with nonnegative entries and zeros on the respective diagonals. In the case that condition (i) or condition (ii) holds and $n > 9$, the optimal value of $QAP(A,B)$ is 0 and an optimal solution can be found in $O(n^2)$ time or $O(n \log n)$ time, respectively:

(i) The matrices A and B have (together) at most $4n-6$ non-zero entries and each of them has at most $n-2$ non-zero entries per row.

(ii) The matrices A and B have (collectively) at most $3n-3$ non-zero entries.

Clearly, for $n \le 9$ the corresponding QAP can be solved in constant time.

Proof (Sketch).
Proof of (i). Consider two undirected weighted graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ whose adjacency matrices are A and B respectively. Obviously, $|V_1| = |V_2| = n$, $\Delta(G_1) < n-1$, $\Delta(G_2) < n-1$ and $|E_1| + |E_2| \le 2n-3$. Bollobás and Eldrige [19] prove that under these conditions, G_2 can be packed into G_1 (with a finite number of exceptions). The proof given in [19] is done by induction on the number of vertices of G_1 and G_2. This proof leads in a natural way to a $O(n^2)$ recursive algorithm for finding a packing. There are at most n recursive iterations and in each of them $O(n)$ constant time graph operations are performed. Before starting the algorithm a preprocessing is needed for sorting the vertices of G_1 with respect to the degree, for computing the connected components of G_2 and for sorting them according to their densities. The preprocessing takes $O(n \log n)$ time. Thus, a packing π of G_2 into G_1 exists and can be found in $O(n^2)$ time. Obviously, the permutation π is an optimal solution to $QAP(A,B)$ as $Z(A,B,\pi) = 0$.
Finally, each of the finitely many exceptions consists of a pair of graphs which have at most 9 vertices. Hence, the corresponding QAPs can be solved in constant time. Clearly, in these cases the optimal value of $QAP(A,B)$ is positive.

Proof of (ii). Again consider two weighted undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with adjacency matrices A and B respectively. Obviously, the inequality $|E_1| + |E_2| \le \frac{3(n-1)}{2}$ holds. Bollobás and Eldrige [19] have shown that in this case

G_2 can be packed into G_1. The proof is again inductive and naturally leads to a recursive algorithm. The recursive procedure is called at most n times and in each call a finite number of constant graph operations are performed. Moreover, the same preprocessing as in Case (i) is needed. Summarizing, a packing, or equivalently, an optimal solution to $QAP(A,B)$ can be found in $O(n \log n)$ time. ∎

At this point, an interesting question would be to investigate the "complexity threshold" of the above type of problem. More precisely, let $f(n)$ denote some non-decreasing function mapping $\mathbb{N}$ to $\mathbb{N}$. Consider a restricted version of the QAP when the input is restricted to pairs (A,B) of symmetric $n \times n$ matrices with nonnegative entries which collectively have at most $2f(n)$ non-zero entries and have zeros on the respective diagonals. For what functions $f(n)$ is this special case of the QAP polynomially solvable? Theorem 4.13 gives a lower bound for this complexity threshold, namely $f(n) = 2n-3$, for $n \in \mathbb{N}$. The following theorem, whose rather technical proof is omitted, gives an upper bound for this threshold.

Theorem 4.14 *Consider the restricted version of $QAP(A,B)$ with symmetric nonnegative coefficients matrices A and B collectively having at most $f(n)$ non-zero entries, where n is the size of the problem. We assume moreover that the matrices A and B have zeros on the respective diagonals. If $f(n) = \Omega(n^{1+\varepsilon})$, for some fixed real $\varepsilon > 0$, then this version of QAP is $\mathcal{NP}$-hard.* ∎

We feel that the above mentioned complexity threshold is much closer to the lower bound given by Theorem 4.13 than to the upper bound given by Theorem 4.14, say around $f(n) = 4n$. However, this is only a conjecture and closing the gap between the above lower and upper bounds remains an open problem.

The next theorem, implicitly proved by Bollobás and Eldrige, suggests that *separately* bounding the number of non-zero entries of the coefficient matrices A and B instead of bounding the sum of these numbers, leads to another complexity threshold scenario.

Theorem 4.15 (Bollobás and Eldrige [19], 1978)
Consider a $QAP(A,B)$, where A and B are two symmetric $n \times n$ matrices with nonnegative entries and zeros on the respective diagonal. Assume that the matrices A and B have at most $2\alpha n$ and $2[(1-2\alpha)/5]n^{3/2}$ non-zero entries, respectively, for some $0 < \alpha < 1/2$. Then, $QAP(A,B)$ is solvable in $O(n^{7/2})$ time and its optimal value equals zero.

Proof (Sketch).
Clearly, the matrices A and B can be viewed as weighted adjacency matrices of two undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, whose edge sets E_1, E_2 fulfill the inequalities $|E_1| \le \alpha n$ and $|E_2| \le [(1-2\alpha)/5]n^{3/2}$, for some $0 < \alpha < 1/2$. Bollobás and Eldrige have shown that under these conditions there exists a packing of G_2 into G_1. Their proof is constructive and yields immediately an algorithm for computing this packing as follows. Begin by assigning the isolated vertices of G_1 to the vertices of

G_2 with largest degrees. Then, update both graphs by deleting the already assigned vertices and the edges incident to them. Consider the updated graphs, say G_1' and G_2', respectively. Assign the isolated vertices of G_2' to vertices of G_1' with largest degrees and again update graphs G_1', G_2' as described above. Denote the updated graphs by G_1'' and G_2'', respectively. Consider an arbitrary embedding of G_2'' into G_1'', that is a one to one mapping of the vertices of G_1'' into the vertices of G_2''. Notice that this embedding is not necessarily a packing. Apply an iterative improvement procedure until a packing results. This procedure gets an arbitrary embedding as input and outputs a new embedding which has a *strictly smaller* number of edge coincidences than the input. This procedure, which can be naturally derived from the proof in [19], runs in $O(n^2)$ time. Considering that this procedure is repeated at most $O(n^{3/2})$ times completes the proof. ∎

Next, let us present a number of results on QAP formulations of packing problems which can be solved in linear time.

Theorem 4.16 *Let A and B be the weighted symmetric $n \times n$ adjacency matrices of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ with nonnegative edge weights and zeros on the respective diagonals. Then, $QAP(A, B)$ is solvable in linear time if one of the following conditions is fulfilled:*

(a) G_1 and G_2 both are trees not isomorphic to the complete bipartite graph (star) $K_{1,n-1}$.

(b) G_1 is a regular graph of degree 2 (all of its vertices have degree 2) and G_2 is a tree. G_2 is neither isomorphic to the star $K_{1,n-1}$ nor isomorphic to a tree resulting from the star $K_{1,n-2}$ when adding one *vertex x to it and substituting* one *of its edges, say (u, v), by two other edges (u, x) and (x, v).*

(c) Both G_1 and G_2 are regular graphs of degree 2.

(d) Both G_1 and G_2 have bandwidth 2.

Proof (Sketch).
Proof of (a). In [88], Hedetniemi, Hedetniemi and Slater prove the existence of a packing of G_2 into G_1, in the case that G_1 and G_2 are weighted graphs which satisfy condition (a). The proof is straightforward and is done by induction. It immediately leads to an $O(n)$ time algorithm for finding a packing of G_2 into G_1, or equivalently, an optimal solution to $QAP(A, B)$ with optimal value equal to 0.

Proof of (b), (c). If the graphs G_1 and G_2 on n vertices, $n > 6$, fulfill condition (b) or (c), there exists a packing of G_2 into G_1, as shown by Metelski [128]. The proof is constructive and leads to an $O(n)$ time algorithm for the construction of such a packing. Clearly, there is a finite number of pairs of graphs with at most 6 vertices and for each of them the corresponding QAP can be solved in constant time.

Proof of (d). Azarionok [6] proves the existence of a packing of G_2 into G_1, for graphs G_1 and G_2 on n vertices, $n > 16$, fulfilling condition (d). He gives a concrete embedding of G_1 into G_2 and proves that it is a packing. This embedding can be constructed by performing $O(n)$ elementary operations. Clearly, QAPs corresponding to packing problems on graphs with at most 16 vertices can be solved in constant time. ∎

The following theorem is based on another polynomiality result on packing problems and is due to Sauer and Spencer [164].

Theorem 4.17 (Sauer and Spencer [164], 1978)
Let A and B be two symmetric $n \times n$ matrices with nonnegative entries and zeros on the respective diagonals. Denote by α and β the maximum number of non-zero entries per row for the matrices A and B, respectively. In the case that $2\alpha\beta < n$, the problem $QAP(A, B)$ is polynomially solvable and its optimal value equals 0.

Proof. Consider two weighted graphs G_1 and G_2 on n vertices, such that A and B are their weighted adjacency matrices, respectively. Clearly, $2\alpha\beta < n$ implies $2\Delta(G_1)\Delta(G_2) < n$, where $\Delta(G_i)$ is the maximum degree of the vertices of G_i, $i = 1, 2$. Sauer and Spencer [164] show that for G_1 and G_2 fulfilling this condition there exists a packing π of G_2 into G_1. This is proven by considering an embedding of G_2 into G_1 with a minimum number of edge coincidences. Under the assumption that this embedding is not a packing, Sauer and Spencer derive a new embedding of G_2 into G_1 with a strictly smaller number of edge coincidences. This contradiction proves the existence of a packing of G_2 into G_1.
Using the idea of the above mentioned construction, an improvement algorithm for finding a packing of G_2 into G_1 can be derived. Begin with an arbitrary embedding and construct a new embedding with strictly smaller number of edge coincidences (in the same way as Sauer and Spencer do). A straightforward complexity analysis shows that this improving step can be performed in $O(n^{3/2})$ time. Repeat this step until a packing is constructed. Clearly, the number of repetitions cannot exceed $\min\{|E_1|, |E_2|\}$ which, under the conditions of the theorem, is $O(n^{3/2})$. Summarizing, finding a packing of G_1 into G_2, or equivalently, an optimal solution to $QAP(A, B)$, takes $O(n^3)$ time. ∎

There is another interesting result in [164] which looks as follows, when translated in the QAP language:

Theorem 4.18 (Sauer and Spencer [164], 1978)
Let A and B be two symmetric $n \times n$ matrices with nonnegative entries and zeros on the respective diagonals. Denote by α and β the number of non-zero entries in A and B, respectively. If $\alpha\beta < 2n(n-1)$, the optimal value of $QAP(A, B)$ equals zero.

Proof. Similarly as in the proof of the previous theorem, consider two weighted graphs G_1 and G_2 on n vertices such that A and B are their weighted adjacency

matrices, respectively. Clearly, $\alpha\beta < 2n(n-1)$ implies $2|E_1||E_2| < n(n-1)$. Sauer and Spencer [164] show that for G_1 and G_2 fulfilling this condition there exists a packing π of G_2 into G_1. Obviously, $Z(A, B, \pi) = 0$ and this completes the proof. ∎
It is worthy to notice that the computational complexity of the QAP version considered in Theorem 4.18 is an open problem. In this case, the proof of the existence of a packing of G_2 into G_1 uses some probabilistic arguments. Exploiting this proof for constructing an algorithm which computes a packing seems to be impossible.

We conclude this section with an $\mathcal{NP}$-hardness result, which additionally shows that the nonnegativity condition on the edge weights in Theorem 4.16 (a) is essential.

Theorem 4.19 (Korneenko [106], 1982)
The special case of QAP(A, B), where A and $-B$ are the adjacency matrices of two unweighted trees, respectively, is $\mathcal{NP}$-hard. ∎

4.4 Special cases of the GQAP

Most of the results in this section concern versions of the GQAP where the coefficient matrices A and B are (symmetric) weighted adjacency matrices of two *isomorphic* (undirected) graphs. Moreover, the minimization occurs over *the set of isomorphisms* between these graphs. Throughout this section, *the set of isomorphisms between two isomorphic graphs* G_1 and G_2, is denoted by $\mathcal{I}(G_1, G_2)$. Clearly, $\mathcal{I}(G_1, G_2) \subseteq S_n$.

Such a GQAP version of this kind was initially introduced by Christofides and Gerrard [41] in 1976 and was termed as *"a graph theoretical formulation of the QAP"*. Given two arbitrary graphs G_1 and G_2, the set $\mathcal{G}(G_1, G_2)$ consists of the subgraphs of G_2 which are isomorphic to G_1. For any $G \in \mathcal{G}(G_1, G_2)$ the set of isomorphisms between G_1 and G is denoted by $\mathcal{I}(G_1, G)$. Finally, the weighted adjacency matrices of G_1 and G are denoted by $A = (a_{ij})$ and $B^G = (b^G_{ij})$ respectively. Christofides and Gerrard considered a generalized version of the GQAP formulated as follows:

$$\min_{G \in \mathcal{G}(G_1,G_2)} \left\{ \min_{\pi \in \mathcal{I}(G_1,G)} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i)\pi(j)} b^G_{ij} \right\} , \tag{4.1}$$

where n is the cardinality of the vertex set of G_1.
If G_1 and G_2 are isomorphic we have $\mathcal{G}(G_1, G_2) = \{G_2\}$ and the problem formulated in (4.1) is the following GQAP:

$$\min_{\pi \in \mathcal{I}(G_1,G_2)} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij} \tag{4.2}$$

where $B = (b_{ij})$ is the adjacency matrix of G_2 and both G_1 and G_2 are graphs on n vertices.

Christofides and Gerrard identified trivial cases of problem (4.2), where the cardinality of $\mathcal{I}(G_1, G_2)$ grows polynomially with the size n of the problem. As simple examples consider the cases when both graphs G_1 and G_2 are chains, cycles or wheels. Dealing with the non-trivial cases, where $|\mathcal{I}(G_1, G_2)|$ grows exponentially with the size of the problem, the strongest result in [41] is stated by the following theorem:

Proposition 4.20 (Christofides and Gerrard [41], 1976)
Let T_1 and T_2 be two isomorphic trees on n vertices with arbitrary weights on the edges and let A and B be the weighted (symmetric) adjacency matrices of T_1 and T_2, respectively. Then the following GQAP

$$\min_{\pi \in \mathcal{I}(T_1, T_2)} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij} \tag{4.3}$$

is polynomially solvable.

Proof (Sketch).
The proof consists of a dynamic programming scheme for polynomially solving problem (4.3). Actually, the authors construct a dynamic programming scheme for the special case when both trees are *multistars*. Then, they notice that this scheme can be modified in order to work also for general trees. We shortly describe this dynamic programming approach for general trees T_1 and T_2.

First, we define *roots* in T_1 and T_2 and assign a *level* to each of their vertices, respectively. This is done by applying a *shelling* procedure to each of T_1 and T_2, as follows. Assign level 0 to all leaves of the tree. Delete all level-0 vertices together with their incident edges. Find the leaves in the remaining tree and assign level 1 to all of them. Then delete all level-1 vertices and the edges incident to them. Repeat this step by increasing the value of the level assigned to the leaves of the current tree by one. Continue until all vertices are labeled. The last level set contains either one or two vertices. In the first case, this uniquely determined vertex is the root of the tree. In the other case one additional vertex, subdiving the last edge, is added as root. The level of the root is called *level of the tree*.

From the above construction follows that an isomorphism between two isomorphic trees T_1 and T_2 maps their roots to each other. Therefore, isomorphic trees have equal levels, say ℓ. Moreover, it is easily seen that each isomorphism between T_1 and T_2 preserves the level of the vertices, i.e., each vertex of T_1, is mapped to a vertex of T_2 with the same level. Once a root is defined, fathers and sons are also defined, as usually. The father of vertex i is denoted by $p(i)$ and the set of sons of i is denoted by $S(i)$.

After this preprocessing the dynamic programming scheme works as follows. Begin by computing the *cost* $c_{ij} = 2a_{ip(i)}b_{jp(j)}$ of mapping vertex i of T_1 to vertex j of T_2, for any i and j of level 0 in T_1 and T_2, respectively. Assume that we have recursively computed such costs for all pairs of vertices with level smaller then k and we want to compute them for vertices with level equal to k. An isomorphism between T_1 and T_2 may map a vertex i of T_1 with level k into a vertex j of T_2 with level k only if $|S(i)| = |S(j)|$. Let i and j be two such vertices in T_1 and T_2, respectively. Consider an $|S(i)| \times |S(j)|$ matrix $C^{(i,j)}$ whose entries $c_{kl}^{(i,j)}$ are the costs of mapping k to l for any pair (k,l) with $k \in S(i)$, $l \in S(j)$. Solve the linear assignment problem with coefficient matrix $C^{(i,j)}$ and denote its optimal value by $\Lambda(i,j)$. Now the cost c_{ij} of mapping i to j is given by the following formula

$$c_{ij} = 2a_{ip(i)}b_{jp(j)} + \Lambda(ij)$$

Apply this process until the cost $c(r_1, r_2)$ is calculated. This is the optimal value of the considered GQAP. The optimal solution corresponding to this value can be found by backtracking in the usual dynamic programming manner.

This algorithm runs in polynomial time. Indeed, its time complexity is dominated by the time needed to solve the linear assignment problems. For assigning the vertices of level t, $1 \le t \le \ell - 1$, in T_1 to the vertices of level t in T_2 at most n_t^2 linear assignment problems are solved, where n_t is the number of vertices of level t in T_1 (and also in T_2). The size of these linear assignment problems is smaller than or equal to $n-1$. Thus, a straightforward upper bound for the time complexity of this step is $n_t^2 n^3$. Summing up the time cost of all steps of the dynamic programming scheme, we get the following inequality which completes the proof:

$$\sum_{t=1}^{\ell} n_t^2 n^3 \le n^3 \left(\sum_{t=1}^{\ell} n_t \right)^2 \le n^5 \qquad \blacksquare$$

Rendl [150] investigated generalizations of the above result to vertex series parallel digraphs. He investigated the GQAP on *minimal vertex series parallel* (MVSP) digraphs showing that this problem is $\mathcal{NP}$-hard. His proof for the $\mathcal{NP}$-hardness of the GQAP on MVSP digraphs works for an even more general class of graphs. So we get the following theorem:

Theorem 4.21 (Rendl [150], 1986)
Consider a class $\mathcal{K}$ of digraphs such that for all $n \in \mathbb{N}$, $n \ge 2$, there exists a graph $G \in \mathcal{K}$ which contains the complete bipartite digraph $K_{n,n}$ as an induced subgraph. Let A and B be the weighted adjacency matrices of two isomorphic graphs G_1 and G_2 in $\mathcal{K}$. Then, the problem GQAP$(A, B, \mathcal{I}(G_1, G_2))$ is $\mathcal{NP}$-hard.

Proof. In [150] it is shown that a QAP of size n can be polynomially reduced to a *GQAP*$(A, B, \mathcal{I}(G_1, G_2))$, where both G_1 and G_2 are weighted complete bipartite digraphs isomorphic to $K_{n,n}$, and A and B are their weighted adjacency matrices, respectively. (The reader should notice that $K_{n,n}$ is a MVSP digraph). $\blacksquare$.

Remark Another class of graphs which fulfill the conditions of Theorem 4.21 is the class of cographs. (The complete bipartite graphs $K_{n,n}$, $n \in \mathbb{N}$, are cographs.) Other examples of graph classes for which analogous $\mathcal{NP}$-hardness results can be derived are, for example, the classes of interval graphs, chordal graphs and split graphs.

Although problem (4.2) on MVSP digraphs is generally $\mathcal{NP}$-hard, it becomes polynomially solvable when restricted on specific subclasses of MVSP digraphs, as shown by Rendl in [150].

Proposition 4.22 (Rendl [150], 1986)
Let G_1 and G_2 be two isomorphic minimal vertex series parallel (MVSP) digraphs with arbitrary weights on the edges, and let A and B be their weighted adjacency matrices, respectively. If none of G_1 and G_2 contains the complete bipartite digraph $K_{2,2}$ as (vertex) induced subgraph, then $GQAP(A, B, \mathcal{I}(G_1, G_2))$ can be solved in polynomial time.

Proof (Sketch).
The proof of this result makes use of the *canonical decomposition trees* of G_1 and G_2. Rendl observes that an isomorphism between G_1 and G_2 leads to an isomorphism between their canonical trees, say T_1 and T_2, respectively. Moreover, this isomorphism between T_1 and T_2 preserves the labels on the nodes of this trees. (A short information on MVSP digraphs and their canonical decomposition trees is given in Subsection 1.3.4. For a detailed information the reader is referred to [180] and [150].) Thus, the canonical decomposition trees of two isomorphic MVSP digraphs are isomorphic. Under these conditions, it takes some additional technicalities to transform (in a certain sense) the GQAP on the given graphs into a GQAP on the corresponding canonical decomposition trees. The main idea is that if an isomorphism $\pi \in \mathcal{I}(G_1, G_2)$ maps vertex v of tree T_1 to vertex v' of tree T_2, then it maps the subtree of T_1 rooted at v to the subtree of T_2 rooted at v_1. If for all vertices v of T_1, the assignment problems related to subtrees rooted at sons of v are somehow *independent*, then a dynamic programming approach similar to that described in the proof of Proposition 4.20 can be applied in polynomial time. Otherwise this approach does not work. The "nice" cases occur if and only if tree T_1 (and T_2, as well) does not contain two sons labeled by P of a father labeled by S. Finally, it is not difficult to show that the canonical decomposition tree of a MVSP digraph has this property if and only if the digraph itself does not contain $K_{2,2}$ as (vertex) induced subgraph. ■

The result of Theorem 4.22 can be generalized for *edge* series parallel digraphs. These problems are solvable in polynomial time by standard techniques. Other easy cases may arise from planar graphs or partial k-trees.

The following results due to Christofides and Gerrard show that relaxing the isomorphism condition in Proposition 4.20 leads to $\mathcal{NP}$-hard problems. Namely, the generalized problem (4.1) is $\mathcal{NP}$-hard in the case that G_1 is a tree and G_2 is a complete graph. solvable problems.

Proposition 4.23 (Christofides and Gerrard [41], 1976)
Let A be the weighted adjacency matrix of a tree G_1 and B be the weighted adjacency matrix of a complete graph G_2. Then, the generalized GQAP is $\mathcal{NP}$-complete. $\blacksquare$

For special trees we get, however:

Proposition 4.24 (Christofides and Gerrard [41], 1976)
Let $A = (a_{ij})$ be the weighted adjacency matrix of a star G_1 and $B = (b_{ij})$ be the weighted adjacency matrix of a complete graph G_2. Then the generalized GQAP (4.1) is solvable in $O(n^2 \log n)$ time, where n is the number of vertices of G_1 and G_2.

Proof. Let the vertex sets of G_1 and G_2 be given as $\{u_1, u_2, \ldots, u_n\}$ and $\{v_1, v_2, , \ldots, v_n\}$, respectively. The weights on the edges of both graphs are denoted by w for simplicity, e.g. $w(v_i, v_j)$, $w(u_i, u_j)$. W.l.o.g., denote the central vertex of the star G_1 by u_1. First consider all permutations π which map k to 1, for some prespecified $1 \le k \le n$. (That is, fix the vertex v_k of G_2 to which u_1 is embedded.) Find π_k which minimizes the double sum $\sum_{i=1}^{n} \sum_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij}$ over all π as above. It is easily seen that this minimization problem is equivalent to the minimization of the scalar product of the vectors $(w(u_1, u_j))^{\pi}$, $1 \le j \le 2$, and $(w(v_k, v_j))$, $j \ne k$, over all π. The latter problem can be solved in $n \log n$ time as stated by Proposition 1.1.

This procedure is repeated n times for $1 \le k \le n$, considering all possible embeddings of the central vertex u_1. Then, choose one permutation which yields the minimum objective function value among π_k, $1 \le k \le n$, derived as above. Clearly, this is an optimal solution of the considered GQAP and it takes $O(n^2 \log n)$ elementary operations to compute it. $\blacksquare$

We conclude this section with an $\mathcal{NP}$-hardness result on GQAPs which are not related to isomorphic graphs. Namely, we show that minimizing the objective function of the quadratic assignment problem over pyramidal permutations is $\mathcal{NP}$-hard. At this point, the QAP again seems to be "more difficult" than the TSP. It is well known that an optimal pyramidal tour on n cities can be found in $O(n^2)$ time (see, for example, [116]). First, let us define a pyramidal permutation.

Definition 4.3 *A permutation $\pi \in \mathcal{S}_n$ is called pyramidal if there exists an $i_0 \in \{1, 2, 3, \ldots, n-1, n\}$ such that either $\pi(i) < \pi(i+1)$ for $i < i_0$ and $\pi(i) < \pi(i-1)$ for $i > i_0$, or $\pi(i) > \pi(i+1)$ for $i < i_0$ and $\pi(i) > \pi(i-1)$ for $i > i_0$.*
The subset of $\mathcal{S}_n$ consisting of the pyramidal permutations of $\{1, 2, \ldots, n\}$ is denoted by $\mathcal{Y}_n$.

Theorem 4.25 *Let A and B be two arbitrary $2n \times 2n$ matrices. The following GQAP is $\mathcal{NP}$-hard:*

$$\min_{\pi \in \mathcal{Y}_{2n}} \sum_{i=1}^{2n} \sum_{j=1}^{2n} a_{\pi(i)\pi(j)} b_{ij} \tag{4.4}$$

Proof. The proof basically consists on showing the existence of an $\mathcal{NP}$-hard version of the QAP which possesses a pyramidal optimal solution. Such a problem is $QAP(A,B)$ with $A \in$ Product(Sym, Even) and $B = (b_{ij})$ a $2n \times 2n$ matrix defined by

$$b_{ij} = \begin{cases} 1 & \text{if} \quad i,j \in \{1,2,\ldots,n\} \text{ or } i,j \in \{n+1,\ldots,2n\} \\ -1 & \text{otherwise} \end{cases} \tag{4.5}$$

The $\mathcal{NP}$-hardness of this problem follows from Theorem 3.17. Indeed, $B \subseteq_\pi$ Chess and therefore $QAP(A,B)$ is equivalent to Product(Sym,Even)×Chess. In the proof of Theorem 3.17 it has been shown that the EquiPartition problem can be reduced to Chess×Product(Sym,Odd). Thus, $QAP(A,B)$, with A and B as described above, is $\mathcal{NP}$-hard. Further, we show that $QAP(A,B)$ possesses a pyramidal optimal solution. Let (α_i) be the generating vector of matrix B. Then, for all $\pi \in \mathcal{S}_{2n}$, the objective function of $QAP(A,B)$ can be written as follows:

$$Z(A,B,\pi) = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{\pi(i)}\alpha_{\pi(j)} + \sum_{i=n+1}^{2n}\sum_{j=n+1}^{2n} \alpha_{\pi(i)}\alpha_{\pi(j)} - 2\sum_{i=1}^{n}\sum_{j=n+1}^{2n} \alpha_{\pi(i)}\alpha_{\pi(j)}$$

The last equality implies that two permutations π_1 and π_2 which fulfill the condition "for all $1 \le i \le n$, there exists a $1 \le j \le n$ with $\pi_1(i) = \pi_2(j)$", yield the same objective function value, i.e., $Z(A,B,\pi_1) = Z(A,B,\pi_2)$.
Consider now an optimal solution π_0 to $QAP(A,B)$. Let $\{i_1,i_2,\ldots,i_n\} = \{1,2,\ldots,n\}$ such that $\pi_0(i_1) < \pi_0(i_2) < \ldots < \pi_0(i_n)$ and $\{i_{n+1},i_{n+2},\ldots,i_{2n}\} = \{n+1,n+2,\ldots,2n\}$ such that $\pi_0(i_{n+1}) > \pi_0(i_{n+2}) > \ldots > \pi_0(i_{2n})$. Consider a new permutation $\pi' \in \mathcal{S}_{2n}$ defined by $\pi' = \langle \pi_0(i_1), \pi_0(i_2), \ldots, \pi_0(i_{2n}) \rangle$. Obviously, permutation π' is pyramidal and moreover, $\{\pi_0(1),\pi_0(2),\ldots,\pi_0(n)\} = \{\pi'(1),\pi'(2),\ldots,\pi'(n)\}$. Therefore, according to the above remark, $Z(A,B,\pi_0) = Z(A,B,\pi')$. Hence π' is an optimal pyramidal solution to $QAP(A,B)$.

Thus, we have proven that $GQAP(A,B,\mathcal{Y}_{2n})$ is equivalent to $QAP(A,B)$, where $A \in$ Product(Sym, Even) and B is given by (4.5). The $\mathcal{NP}$-hardness of the later problem implies the $\mathcal{NP}$-hardness of problem (4.4) and this completes the proof. ■

Part III

A Generalization of the QAP and Related Problems

Chapter 5

On the Biquadratic Assignment Problem (BiQAP)

In this chapter we investigate a natural generalization of the QAP, namely the *biquadratic assignment problem*, shortly denoted by BiQAP. In the BiQAP, a weighted sum of products of four variables x_{ij} is to be minimized, subject to assignment constraints on the variables x_{ij}. The investigation of BiQAPs is motivated by a problem arising in the VLSI design which can be mathematically modeled as a BiQAP. This practical application of BiQAPs is discussed in detail in the next section.

Consider two arrays $A = (a_{ijkl})$ and $B = (b_{mpst})$ of n^4 elements each and the variables x_{ij}, $i, j = 1, \ldots, n$. Then, the BiQAP can be mathematically written as follows:

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{m=1}^{n}\sum_{p=1}^{n}\sum_{s=1}^{n}\sum_{t=1}^{n} a_{ijkl} b_{mpst} x_{im} x_{jp} x_{ks} x_{lt} \tag{5.1}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \,, \qquad j = 1, 2, \ldots, n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \,, \qquad i = 1, 2, \ldots, n$$

$$x_{ij} \in \{0, 1\}.$$

Note that the constraints of the BiQAP are the usual assignment constraints. Thus, the variables x_{ij} can be represented by a permutation of the set $\{1, 2, \ldots, n\}$. In this case we obtain the following alternative formulation of the BiQAP:

$$\min_{\pi \in \mathcal{S}_n} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} a_{\pi(i)\pi(j)\pi(k)\pi(l)} b_{ijkl} \,, \tag{5.2}$$

where $\mathcal{S}_n$ denotes as usually the set of all permutations of $\{1, 2, \ldots, n\}$. The formulation in (5.2) clearly shows that the BiQAP is a generalization of the quadratic assignment problem (QAP) to which it owes its name. Also the BiQAP terminology is very similar to that used for QAPs. The notations *BiQAP*(A, B), $Z(A, B, \pi)$ and the terms *optimal solution*, *optimal value*, *size of the BiQAP*, *equivalent BiQAPs*, have a completely analogous meaning to that of the corresponding terms used for QAPs. Moreover, *a QAP and BiQAP* are said to be *equivalent* in the case that once we know an optimal solution to one of the problems, we may construct an optimal solution to the other one.

The rest of this chapter is organized as follows. In the first section we describe a practical application of the BiQAP which arises in the field of VLSI synthesis. In Section 5.2 we present two alternative formulations of the BiQAP, in particular mixed integer programming formulations. It can be easily shown that the strong relationship between QAPs and BiQAPs implies the $\mathcal{NP}$-completeness of BiQAP. Hence, it seems unlikely to find a polynomial time algorithm for BiQAPs and the implicit enumeration is the only potentially available approach to optimally solve the problem. In this frame, the computation of lower bounds becomes relevant. In Section 5.3 we introduce a Gilmore-Lawler type lower bound for BiQAPs. Similarly as in the case of QAPs, these bounds can be strengthened by using reduction methods, analogous to those used for QAPs. We shortly introduce two of these methods and present the outlet of an lower bounding algorithm for BiQAPs involving such reduction methods. Then, in Section 5.4, a procedure for generating BiQAP instances with known optimal solution is presented. This procedure is very much in the flavor of analogous procedures for QAPs proposed by Palubeckis [138] and Li and Pardalos [117]. In particular, it generalizes and uses one of the methods presented in [117].
Clearly the benefit such BiQAP instances concerns testing the quality of lower bounds when using test problems with known optimal solution. In this frame, we report on our computational experience with lower bounds for BiQAP instances with known optimal value, generated by the above mentioned procedure. Finally, in Section 5.5, we analyze the asymptotic behavior of BiQAPs showing that the similarity between QAPs and BiQAPs extends also to this aspect. Namely, it turns out that under certain probabilistic constraints, the ratio between the "worst" and the "best" objective function value is arbitrarily close to one with probability tending to one as the size of the problem tends to infinity.

5.1 An application of BiQAPs arising in the VLSI synthesis

Almost all VLSI circuits are so-called sequential circuits. A sequential circuit is an interconnection of storage elements, so called flip-flops, and logic gates. There are two types of sequential circuits, synchronous and asynchronous circuits. In the following we deal only with synchronous sequential circuits that employ storage elements which are allowed to change their value only at discrete instants of time.

Sequential circuits are most often modeled by using finite state machines (FSMs). An FSM is a mathematical model of a system with discrete inputs, discrete outputs and a finite number of internal configurations or states. FSMs are commonly represented by state transition tables which describe outputs and next states as a function of inputs and present states. (Note that there is no dependence on previous states.) If the state of an FSM changes from s to s', we say that a transition from s to s' takes place and write $s \rightarrow s'$ for short.

The state of a sequential circuit is represented by the states of its flip-flops. A flip-flop is a storage device capable of storing one bit of information. It maintains its binary state (equal to 0 or 1) until directed by a clock pulse to switch states. The difference among various types of flip-flops is in the number of inputs they possess and in the manner in which the inputs affect the binary state. Two simple types of flip-flops with a single input are the D (data) flip-flop and the T (toggle) flip-flop. The differences between these two types are as follows: For the D flip-flop the next state is always equal to the input and does not depend on the present state. This does not hold any longer for the T flip-flop where the next state is obtained by evaluating the exclusive-OR (XOR) function with the input and the present state as arguments. (For more details about flip-flops and sequential circuits in general the reader is referred to Mano [125].) The T flip-flop turns out to be more useful in practice than the D flip-flop, but, as we will see below, it is also more difficult to deal with them.

The first step of the procedure for designing a sequential circuit consists of translating the circuit specifications obtained from the practical application into a state transition table. In the next step we try to find an encoding of the internal symbolic states of the FSM as binary strings such that the eventual implementation after encoding and applying logic minimization is of minimum size. For instance, if the sequential circuit is to be implemented in programmable logic array (PLA) form, then we wish to minimize the number of product terms (or the area) of the resulting PLA.

The state encoding problem is an extremely challenging problem. Most algorithms from the literature focus on area minimization (see e.g. Ashar, Devadas and Newton [3] and Eschermann [54]), but the encoding has also a profound effect on the testability and performance of the resulting implementation (for some work on testability aspects see Eschermann and Wunderlich [55]).

Even if the set of binary codes used to encode the n internal states is fixed in advance, a brute force enumerative approach to the state encoding problem would require $O(n!)$ calls of a logic minimization program. This method is obviously infeasible for practical sizes of n.

Almost all newer state encoding algorithms for two-level (binary) logic follow a two phase strategy first proposed by De Micheli, Brayton and Sangiovanni-

Vincentelli [131]. In the first phase the states of the FSM are encoded symbolically in the following way: The i-th state, $i = 1, 2, \ldots, n$, is encoded by a 0-1 string which has a 1 at position i and 0's elsewhere. Then a logic minimization program is applied to this symbolic representation. The aim of this phase is to generate conditions on the relationships between the codes for different states. The actual form of these constraints depends on the technology and the types of elements which are used to implement the sequential circuit. This issue will be discussed below.

Given the coding constraints obtained in the first phase, it is the task of the second phase to find an encoding of the states which satisfies as many of the coding constraints obtained in the first phase as possible, while using a much smaller encoding length than in the symbolic encoding. (The symbolic encoding uses n bits and hence requires a large number of flip-flops to represent the state of the system.)

What remains to be discussed is the manner how the coding constraints are generated in the first step. Let us start with a simple case before proceeding to the description of the case which leads to a biquadratic assignment problem.

Suppose that all employed flip-flops are D flip-flops. We divide the transitions into groups where two transitions, say $t_1 : s_1 \to s_3$ and $t_2 : s_2 \to s_4$ belong to the same group if they are induced by the same input and produce the same output. In the case of D flip-flops two such transitions can be merged, i.e. implemented by a single product term of PLA, if $s_1 \neq s_2, s_3 = s_4$ and if the states s_1 and s_2 are assigned adjacent codes, i.e. codes with Hamming distance 1. The problem of determining an encoding which satisfies as many of these coding constraints as possible leads to a quadratic assignment problem. (For details see Eschermann and Wunderlich [55].)

Now let us turn to the case where all flip-flops are T flip-flops. Let $f(s)$ denote the code of state s and $d(c, c')$ the Hamming distance of the codes c and c'. Due to the specific properties of T flip-flops, two transitions $t_1 : s_1 \to s_3$ and $t_2 : s_2 \to s_4$ which belong to the same group can be merged if the following three constraints are satisfied:

1. $s_i \neq s_j$ for all $i \neq j$, $i, j = 1, 2, 3, 4$, i.e., the four involved states are pairwise disjoint,
2. $d(\, f(s_1), f(s_2)\,) = 1$, i.e., s_1 and s_2 are assigned adjacent codes, and
3. $f(s_1) \oplus f(s_3) = f(s_2) \oplus f(s_4)$, where $\oplus$ denotes the bitwise exclusive-OR (XOR) operation.

Note that condition (ii) already occurred for D flip-flops while (iii) ensures that the input to the T flip-flops is the same for both transitions. (Recall that for a T flip-flop the input is equal to the XOR of the value representing its present state and the value representing its next state.)

A mathematical model of this problem can be obtained as follows: Let S be the set of states and C be the set of codes which we would like to use for the state encoding. For simplicity we assume $|S| = |C| = n$ and number the states and the codes from 1 to n, i.e. $S = \{s_1, s_2, \ldots, s_n\}$ and $C = \{c_1, c_2, \ldots, c_n\}$. (Usually C is chosen as subset of the set of all 0-1 strings of length $\lceil \log_2 n \rceil$.)

Furthermore the following notational conventions turn out to be useful. We define a set $T \subseteq \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\}$, where $(i, j) \in T$ if and only if the transition table contains a transition $s_i \rightarrow s_j$ from state s_i to state s_j. Two pairs $(i, j) \in T$ and $(k, l) \in T$ are said to be mergeable, if i, j, k, l are pairwise disjoint and if the transitions $s_i \rightarrow s_j$ and $s_k \rightarrow s_l$ belong to the same group (i.e., are induced by the same input and produce the same output). For two mergeable pairs (i, j) and (k, l) we write $(i, j) \sim (k, l)$ for short.

Now two four dimensional arrays of n^4 elements each are defined. The first array $A = (a_{ijkl})$, $i, j, k, l = 1, 2, \ldots, n$, is related to the states and the transitions. The entries of A are given by

$$a_{ijkl} = \begin{cases} 1 & \text{if } (i,j) \in T,\ (k,l) \in T \text{ and } (i,j) \sim (k,l) \\ 0 & \text{otherwise.} \end{cases}$$

The second array $B = (b_{mpst})$, $m, p, s, t = 1, 2, \ldots, n$, is related to the codes. Its entries are given by

$$b_{mpst} = \begin{cases} 0 & \text{if } c_m \oplus c_p = c_s \oplus c_t \text{ and } d(c_m, c_s) = 1 \\ 1 & \text{otherwise} \end{cases}$$

where $\oplus$ again denotes the XOR operation.

We note that $a_{ijkl} = 1$ exactly when the two transitions $s_i \rightarrow s_j$, $s_k \rightarrow s_l$ could potentially be merged, i.e., can be merged if the involved states are encoded appropriately. Furthermore, we have $b_{mpst} = 0$ exactly when the codes c_m, c_p, c_s and c_t satisfy the coding constraints (ii) and (iii).

It is straightforward to see that the state encoding problem can be formulated as BiQAP(A,B) where the coefficient arrays A and B are as defined above. In this case we want to find a permutation $\pi_0 \in \mathcal{S}_n$ which minimizes the following objective function

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} a_{\pi(i)\pi(j)\pi(k)\pi(l)} b_{ijkl} \tag{5.3}$$

over all permutations $\pi \in \mathcal{S}_n$.

For the VLSI synthesis problem $\pi(i)$ denotes the index of the state to which code c_i assigned, i.e., code c_i is a assigned to state $s_{\pi(i)}$. Hence the term $a_{\pi(i)\pi(j)\pi(k)\pi(l)} b_{ijkl}$ is equal to 1 exactly when the two transitions $s_{\pi(i)} \rightarrow s_{\pi(j)}$ and $s_{\pi(k)} \rightarrow s_{\pi(l)}$ could potentially be merged, but this merging step cannot be performed because of the codes of the states $s_{\pi(i)}, s_{\pi(j)}, s_{\pi(k)}$ and $s_{\pi(l)}$.

Therefore minimizing the sum (5.3) minimizes the number of the transitions which cannot be merged because of the chosen state encoding. This is exactly the objective of the state encoding algorithm described above.

5.2 Different formulations for BiQAPs

The mathematical problem described in the introduction is a combinatorial optimization problem which can be put in rather different forms. Most generally we can state it in the form

$$\min_{\pi \in \mathcal{S}_n} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} d_{\pi(i)i\pi(j)j\pi(k)k\pi(l)l} \tag{5.4}$$

where $\mathcal{S}_n$ denotes as usually the set of all permutations of $\{1, 2, \dots, n\}$. Note that the objective function in (5.2) is a special case of the objective function of problem (5.4). In (5.2) the coefficients $d_{\pi(i)i\pi(j)j\pi(k)k\pi(l)l}$ are split as product of two factors:

$$d_{\pi(i)i\pi(j)j\pi(k)k\pi(l)l} = a_{\pi(i)\pi(j)\pi(k)\pi(l)} b_{ijkl} \, , \quad i, j, k, l = 1, 2, \dots, n. \tag{5.5}$$

In analogy to QAPs we refer to BiQAPs which have property (5.5) as *Koopmans-Beckmann problems*. Without loss of generality we assume that the coefficients $d_{\pi(i)i\pi(j)j\pi(k)k\pi(l)l}$ in (5.4) as well as the coefficients $a_{\pi(i)\pi(j)\pi(k)\pi(l)}$ and b_{ijkl} in (5.5) are nonnegative[1]. We mainly deal with BiQAPs in Koopmans-Beckmann form, but all results in this section hold also for the more general problem (5.4).

Next we derive two linear integer programming formulation for BiQAPs. We start from the formulation in (5.1). and linearize its objective function by introducing $O(n^8)$ new variables y_{ijkl}, $z_{ijklmpst}$, $i, j, k, l, m, p, s, t = 1, 2, \dots, n$.

Theorem 5.1 *The BiQAP (5.1) is equivalent to the following 0-1 linear program:*

$$\min \quad z = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{m=1}^{n}\sum_{p=1}^{n}\sum_{s=1}^{n}\sum_{t=1}^{n} a_{ijkl} \cdot b_{mpst} \cdot z_{ijklmpst} \tag{5.6}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{m=1}^{n}\sum_{p=1}^{n} y_{ijmp} = n^2 \qquad (*)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{m=1}^{n}\sum_{p=1}^{n}\sum_{s=1}^{n}\sum_{t=1}^{n} z_{ijklmpst} = n^4 \qquad (**)$$

$$\begin{aligned} y_{ijmp} + y_{klst} - 2z_{ijklmpst} &\geq 0 && i, j, k, l, m, p, s, t = 1, 2, \dots, n \\ x_{ij} + x_{mp} - 2y_{ijmp} &\geq 0 && i, j, m, p = 1, 2, \dots, n \\ \sum_{i=1}^{n} x_{ij} &= 1 && j = 1, 2, \dots, n \end{aligned}$$

[1]Otherwise we can add a constant to all coefficients of the BiQAP in (5.4) such that all of them become nonnegative. This would results to an increase of the objective function values of all feasible solutions by the same constant, but would not change the optimal solution.

$$\begin{aligned} \sum_{j=1}^{n} x_{ij} &= 1 & i &= 1,2,\ldots,n \\ x_{ij} &\in \{0,1\} & i,j &= 1,2,\ldots,n \\ y_{ijmp} &\in \{0,1\} & i,j,m,p &= 1,2,\ldots,n \\ z_{ijklmpst} &\in \{0,1\} & i,j,k,l,m,p,s,t &= 1,2,\ldots,n \end{aligned}$$

Proof. Let $\{x_{ij}\}_{i,j=1,2,\ldots,n}$ be a feasible solution of (5.1). We define

$$y_{ijmp} = x_{ij} \cdot x_{mp} \text{ and } z_{ijklmpst} = y_{ijmp} \cdot y_{klst}, \quad i,j,k,l,m,p,s,t = 1,2\ldots,n \quad (5.7)$$

Then, it is straightforward to see that

$$\{\{x_{ij}\}_{i,j=1,2,\ldots,n}\ , \{y_{ijmp}\}_{i,j,m,p=1,2,\ldots,n}\ , \{z_{ijklmpst}\}_{i,j,k,l,m,p,s,t=1,2,\ldots,n}\}$$

is a feasible solution of (5.6). Moreover these feasible solution yield the same objective function value for corresponding objective functions.

Conversely let

$$\{\{x_{ij}\}_{i,j=1,2,\ldots,n}\ , \{y_{ijmp}\}_{i,j,m,p=1,2,\ldots,n}\ , \{z_{ijklmpst}\}_{i,j,k,l,m,p,s,t=1,2,\ldots,n}\}$$

be a feasible solution of (5.6). From $x_{ij} + x_{mp} - 2y_{ijmp} \geq 0$ and $x_{ij} \in \{0,1\}$, $x_{mp} \in \{0,1\}$, $y_{ijmp} \in \{0,1\}$ it follows that $x_{ij} = x_{mp} = 1$ whenever $y_{ijmp} = 1$. If we assume that there exist four indices i,j,m,p for which $x_{ij} = x_{mp} = 1$, but $y_{ijmp} = 0$ we get a contradiction to the constraint $(*)$. Therefore $y_{ijmp} = x_{ij}x_{mp}$. Using $(**)$ instead of $(*)$, we can quite similarly show that $z_{ijklmpst} = y_{ijmp}y_{klst}$.

Hence, the values of x_{ij}, $i,j = 1,2,\ldots,n$, appearing in a feasible solution of (5.6) form, in fact, a feasible solution of (5.1). Again, these two feasible solutions yields equal values of the corresponding objective functions. ■

The BiQAP formulation introduced by Theorem 5.1 is a 0-1 linear program with $n^8 + n^4 + n^2$ variables and $n^8 + n^4 + 2n + 2$ constraints. A smaller linearization can be obtained by combining the technique used above with a linearization idea of Kaufman and Broeckx [102]. This leads to a mixed integer program with $2n^4 + n^2$ variables and $2n^4 + 2n + 2$ constraints.

Let us first introduce some notation. For each 4-tuple of indices (i,m,j,p) we consider an array

$$D^{(i,m,j,p)} = \left(d_{kslt}^{(i,m,j,p)}\right)$$

of n^4 elements, $k,s,l,t = 1,2\ldots,n$, defined by

$$d_{kslt}^{(i,m,j,p)} = a_{ijkl} \cdot b_{mpst} \quad k,s,l,t = 1,2\ldots,n\ .$$

For two arrays $D = (d_{ijkl})$ and $Y = (y_{ijkl})$ of n^4 entries each let us define the product DY as:

$$DY := \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} d_{ijkl} \cdot y_{ijkl}$$

Now, we can prove the following theorem:

Theorem 5.2 *The BiQAP (5.1) is equivalent to the following mixed integer linear program:*

$$\min \quad z = \sum_{i=1}^{n}\sum_{m=1}^{n}\sum_{j=1}^{n}\sum_{p=1}^{n} w_{imjp} \tag{5.8}$$

$$\begin{array}{rcll}
\sum_{i=1}^{n} x_{ij} & = & 1 & j = 1,2,\ldots,n \\
\sum_{j=1}^{n} x_{ij} & = & 1 & i = 1,2,\ldots,n \\
\sum_{i=1}^{n}\sum_{m=1}^{n}\sum_{j=1}^{n}\sum_{p=1}^{n} y_{imjp} & = & n^2 & \\
x_{im} + x_{jp} - 2y_{imjp} & \geq & 0 & i,m,j,p = 1,2,\ldots,n \\
c_{imjp}y_{imjp} + D^{(i,m,j,p)}Y - w_{imjp} & \leq & c_{imjp} & i,m,j,p = 1,2,\ldots,n \qquad (*) \\
w_{imjp} & \geq & 0 & i,m,j,p = 1,2,\ldots,n \\
y_{imjp} & \in & \{0,1\} & i,m,j,p = 1,2,\ldots,n \\
x_{im} & \in & \{0,1\} & i,m = 1,2,\ldots,n
\end{array}$$

where

$$c_{imjp} = \sum_{k=1}^{n}\sum_{l=1}^{n}\sum_{s=1}^{n}\sum_{t=1}^{n} a_{ijkl}b_{mpst} \qquad i,m,j,p = 1,2,\ldots,n.$$

Proof. If $\{x_{ij}\}_{i,j=1,2,\ldots,n}$ is a feasible solution of (5.1), then we obtain a feasible solution of (5.8) by defining

$$y_{imjp} = x_{im}x_{jp} \qquad i,m,j,p = 1,2,\ldots,n$$

$$w_{imjp} = y_{imjp}(D^{(i,m,j,p)}Y) \qquad i,m,j,p = 1,2,\ldots,n.$$

It is obvious to see that these solutions yield the same objective function value of the corresponding objective functions.

Conversely let us assume that

$$\{\,\{x_{ij}\}_{i,j=1,2,\ldots,n},\ \{y_{imjp}\}_{i,m,j,p=1,2,\ldots,n},\ \{w_{imjp}\}_{i,m,j,p=1,2,\ldots,n}\,\}$$

is an optimal solution of (5.8). We wish to prove that

$$y_{imjp} = x_{im}x_{jp} \qquad i,m,j,p = 1,2,\ldots,n, \tag{5.9}$$

$$w_{imjp} = y_{imjp}(D_{imjp}Y) \qquad i,m,j,p = 1,2,\ldots,n. \tag{5.10}$$

Then, using only simple calculations, we could find that the set of variables x_{ij}, $i,j = 1,2,\ldots,n$, from the considered optimal solution of (5.8) form an optimal solution of (5.1).

The proof of equality (5.9) is done in precisely the same way as in the preceding theorem. In order to prove the correctness of equality (5.10) we notice that

$D^{(i,m,j,p)}Y \le c_{imjp}$ always holds. Therefore, whenever $y_{imjp} = 0$, the nonnegativity constraints are the only constraints on the variables w_{imjp}. Since we are minimizing the sum of the w_{imjp}, it follows that $w_{imjp} = 0$ whenever $y_{imjp} = 0$.

In the case that $y_{ijmp} = 1$ holds, then the constraint $(*)$ yields

$$w_{imjp} \ge D^{(i,m,j,p)}Y.$$

Again, the fact that we are minimizing the sum of the variables w_{imjp} yields

$$w_{imjp} = D^{(i,m,j,p)}Y = y_{imjp}(D^{(i,m,j,p)}Y)$$

as required. ■

Next let us say a few words about the complexity of the BiQAP which is almost evident. An arbitrary QAP(A,B) of size n, $A = (a_{ij})$, $B = (b_{ij})$, is equivalent to a BiQAP(A′, B′)of size n, $A' = (a'_{ijkl})$, $B' = (b'_{ijkl})$, where:

$$a'_{ijkl} = a_{ij} \text{ and } b'_{ijkl} = b_{ij} \ , \quad i, j, k, l = 1, 2, \ldots, n$$

The following equality hold for each $\pi \in \mathcal{S}_n$ and hence shows that QAP(A,B) and QAP(A′, B′) have a common set of optimal solutions.

$$Z(A', B', \pi) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} a'_{\pi(i)\pi(j)\pi(k)\pi(l)} b'_{ijkl} = n^2 \sum_{i=1}^{n}\sum_{j=1}^{n} a_{\pi(i)\pi(j)} b_{ij} = Z(A, B, \pi)$$

This relationship between QAPs and BiQAPs shows that the computational complexity results derived for QAPs hold also for BiQAPs.

5.3 Bounds for the BiQAP

In this section we derive a Gilmore-Lawler type lower bound for BiQAPs. Moreover, quite similarly as in the case of QAPs, reduction methods can be applied trying to improve the quality of this bound. We have developed two reduction methods for BiQAPs generalizing well know reduction ideas for QAPs (for example, see Burkard [22] and Roucairol [158]). Here we will only shortly describe the basic idea of each of these methods and their real impact on "reducing" the magnitude of the problem coefficients. For an exact description of these methods and a layout of the corresponding algorithms the reader is referred to [26].

Consider BiQAP(A,B) of size n, $A = (a_{ijkl})$ and $B = (b_{ijkl})$, $i, j, k, l = 1, 2 \ldots, n$. For fixed indices i, j, m, p let f_{ijmp} be a lower bound for the following QAP:

$$\min_{\varphi \in \widetilde{\mathcal{S}}_n} \ \sum_{k=1}^{n}\sum_{l=1}^{n} a_{ij\pi(k)\pi(l)} b_{mpkl} \ , \tag{5.11}$$

where $\tilde{\mathcal{S}}_n = \{\pi \in \mathcal{S}_n : \pi(m) = i,\ \pi(p) = j\}$.

Then, for any permutation π of $\{1, 2, \ldots, n\}$, the following inequality holds:

$$f_{\pi(i)\pi(j)ij} \le \sum_{k=1}^{n} \sum_{l=1}^{n} a_{\pi(i)\pi(j)\pi(k)\pi(l)} b_{ijkl} \tag{5.12}$$

Let $\underline{z}$ be a lower bound for the QAP

$$\min_{\pi \in \mathcal{S}_n} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{\pi(i)\pi(j)ij} \tag{5.13}$$

Summing up side by side in (5.12) over i and j and taking then the minimum of each side over $\pi \in \mathcal{S}_n$ we get

$$\underline{z} \le \min_{\varphi \in \mathcal{S}_n} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{\pi(i)\pi(j)ij} \le \min_{\pi \in \mathcal{S}_n} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} a_{\pi(i)\pi(j)\pi(k)\pi(l)} b_{ijkl}. \tag{5.14}$$

Thus, $\underline{z}$ is a lower bound for BiQAP(A,B). In order to compute $\underline{z}$ we have to compute n^4 lower bounds f_{ijmp} for n^4 Koopmans-Beckmann QAPs, respectively, and finally the lower bound of (5.13). Whereas for Koopmans-Beckmann QAPs either the Gilmore-Lawler bounds or eigenvalue-related bounds can be used, only the latter approach can be applied to the general QAP in (5.13).

Now, what about reduction methods? Similarly as for QAPs, the idea behind reduction methods for BiQAPs is to split the objective function of the given BiQAP instance into a sum of the objective function of another BiQAP instance, with smaller coefficients, and objective functions of lower degree assignment problems (QAPs and LAPs), respectively. This approach is justified by the hope that for lower degree assignment problems tighter bounds can be computed.

We have considered two reduction methods. In the first one we derive new coefficients $\overline{a}_{ijkl}$ and $\overline{b}_{mpst}$, $i, j, k, l = 1, 2, \ldots, n$, related to the coefficients a_{ijkl}, b_{mpst}, $i, j, k, l = 1, 2, \ldots, n$, of the given BiQAP(A,B) as follows:

$$\begin{aligned} a_{ijkl} &= \overline{a}_{ijkl} + a^{(1)}_{ij} + a^{(2)}_{ik} + a^{(3)}_{il} + a^{(4)}_{jk} + a^{(5)}_{jl} + a^{(6)}_{kl}\ , \\ b_{mpst} &= \overline{b}_{mpst} + b^{(1)}_{mp} + b^{(2)}_{ms} + b^{(3)}_{mt} + b^{(4)}_{ps} + b^{(5)}_{pt} + b^{(6)}_{st}\ , \\ & \quad i, j, k, l, m, p, s, t = 1, 2, \ldots, n\ , \end{aligned} \tag{5.15}$$

The coefficients $\overline{a}, \overline{b}, a^{(i)}, b^{(i)}$, $1 \le i \le 6$, are appropriately chosen in order to produce as many zeros as possible in each of the resulting arrays $\bar{A} = (\bar{a}_{ijkl})$ and $\bar{B} = (\bar{b}_{ijkl})$. Moreover, the entries of both arrays $\bar{A}$ and $\bar{B}$ should remain nonnegative. The choice proposed in [26] produces at least n^2 zeros in each of the arrays $\bar{A}$ and $\bar{B}$, whereas the time complexity of the corresponding procedure is $O(n^4)$.

After plugging the new coefficients $\bar{a}$ and $\bar{b}$ in the objective function of the given BiQAP(A,B), the latter splits into a new problem BiQAP($\overline{\text{A}}, \overline{\text{B}}$) (hopefully of a simpler form), 6 constants, 24 linear assignment problems and 18 QAPs. We can sum up the coefficients of the QAPs (LAPs) and form a new QAP (LAP). Thus, the objective function of BiQAP(A,B) is then presented as the sum of a constant with the objective function of BiQAP($\overline{\text{A}}, \overline{\text{B}}$), and the objective functions of a QAP and a LAP, respectively. A lower bound for this decomposed problem results as sum of lower bounds for each of the composing problems. The outlet of an algorithm which computes a Gilmore-Lawler type bound for a such decomposed problem is presented in the next subsection. However, the main point concerns the coefficients of BiQAP($\bar{\text{A}}, \bar{\text{B}}$). As they are smaller then the coefficients of BiQAP(A,B) and at least n^2 of them are equal to zero, one hopes that a lower bound for BiQAP($\bar{\text{A}}, \bar{\text{B}}$), computed as described previously in this section, will be tighter than the corresponding bound for BiQAP(A,B).

In the second reduction method, the largest entry of each of the coefficients arrays A and B is iteratively decreased. This reduction process is applied a fixed number M of times to both arrays A and B. Computational experiments have shown that the control parameter M does not need to be larger than 5, since further reductions do not lead to further improvements. (A similar idea for QAPs has been suggested by Roucairol [158].)

After applying this reduction method, the given problem BiQAP(A,B) is split into a new problem BiQAP($\bar{\text{A}}, \bar{\text{B}}$) and a QAP, not necessarily of Koopmans-Beckmann form. The largest entry of the arrays $\bar{A}$ and $\bar{B}$ are smaller than those of A and B, respectively, whereas the time cost of this reduction is $O(n^4)$. A lower bound to the decomposed problem can again be computed by applying the algorithm presented below.

5.3.1 An algorithm for lower bounds of BiQAPs

In the following we compute a Gilmore-Lawler type lower bound for the so called *general BiQAP* whose objective function consists of a BiQAP part, a QAP part and a LAP part. More precisely, let two arrays A and B with n^4 entries each and three $n \times n$ matrices C, D, E, be given. Consider the following problem

$$\min_{\pi \in \mathcal{S}_n} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} a_{\pi(i)\pi(j)\pi(k)\pi(l)} b_{ijkl} + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{\pi(i)\pi(j)} d_{ij} + \sum_{i=1}^{n} e_{\pi(i)i} + K \right\} , \quad (5.16)$$

where K is a constant. Clearly, in the case that the matrices C, D and E have only zero entries and also $K = 0$, we get a BiQAP as defined in (5.2).

The proposed bounding procedure is summarized by the following algorithm.

Algorithm 5.1 *Lower bound computation for the general BiQAP.*
Input data: Two arrays of n^4 entries, A and B, three $n \times n$ matrices C, D, E and a constant $K \in \mathbb{R}$.
Output data: A lower bound $\underline{z}$ for the general BiQAP (5.16).

Step 1 *Reduce the coefficients of BiQAP(A,B) using some reduction method. Add the coefficients of the resulting lower degree assignment problems to the coefficients of the corresponding problems (with the same degree) involved in (5.16).*

Step 2 *For $i,j,m,p = 1,2,\ldots,n$, set f_{ijmp} to a lower bound of the following GQAP*

$$\min_{\pi \in \widetilde{\mathcal{S}}_n} \sum_{k=1}^{n} \sum_{l=1}^{n} a_{ij\pi(k)\pi(l)} b_{mpkl} \tag{5.17}$$

where $\widetilde{\mathcal{S}}_n = \{\pi \in \mathcal{S}_n : \pi(m) = i,\ \pi(p) = j\}$.

Step 3 *Update $f_{ijmp} := f_{ijmp} + c_{ij} d_{mp}$, $i,j,m,p = 1,2,\ldots,n$.*

Step 4 *Reduce the non-Koopmans-Beckmann QAP with coefficients f_{ijkl} using one of the well-known reduction methods for QAPs. Add the coefficients of the additional resulting LAP to the e-coefficients and the additional resulting constant to K.*

Step 5 *Let α_{ij} be the coefficients the last LAP resulting when deriving the Gilmore-Lawler bound for the QAP with coefficients f_{ijkl}. Update $e_{ij} := e_{ij} + \alpha_{ij}$, $i,j = 1,2,\ldots,n$.*

Step 6 *Solve the LAP with coefficients e_{ij}. Set $\underline{z}$ equal to the corresponding optimal value.*

Step 7 *Set $\underline{z} := \underline{z} + K$. Output $\underline{z}$.*

The reduction in Step 1 can e.g. be performed by applying the above described reduction methods. Recall that an outlet of the respective algorithms can be found in [26]. The reduction in Step 4 can e.g. be performed by applying the methods mentioned in Subsection 2.4.2 and described in detail in [22, 59, 158]. The overall bounding procedure takes $O(n^7)$ time. As we will see in the next section, the quality of these bounds is not very good and becomes worse when the size of the problem increases, a phenomenon already known from QAPs. This is not surprising, since linearizing a power-four function yields already in simpler models a large error.

5.4 BiQAPs with known optimal solution and computational results

A good way to test the quality of lower bounds is to compute them for test instances with known optimal value. Based on the analogous ideas for QAPs, we introduce here a generator of BiQAP instances with known optimal solution. In Section 2.6 we have described in details the generators of Palubeckis [138] and Li and Pardalos [117] for QAPs with known optimal solution. QAP instances generated by the latter are used here as a starting point to construct BiQAP instances with known optimal solution. The coefficients of these BiQAPs are obtained from QAP coefficients by a kind of matrix product. Namely, assume that the coefficients a_{ijkl}, b_{ijkl}, $i,j=1,2,\ldots,n$, of BiQAP(A,B) are obtained as

$$a_{ijkl}=a_{ij}^{(1)}a_{kl}^{(2)}\ ,\quad b_{mpst}=b_{mp}^{(1)}b_{st}^{(2)}\ ,\quad i,j,k,l,m,p,s,t=1,2,\ldots,n\ , \tag{5.18}$$

where $A^{(t)}=\left(a_{ij}^{(t)}\right)$ and $B^{(t)}=\left(b_{ij}^{(t)}\right)$, $t=1,2$, are $n\times n$ matrices.

Clearly, the objective function of BiQAP(A,B) can then be written as follows, for each $\pi\in\mathcal{S}_n$:

$$\begin{aligned} Z(A,B,\pi) &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} a_{\pi(i)\pi(j)}^{(1)}a_{\pi(k)\pi(l)}^{(2)}b_{ij}^{(1)}b_{kl}^{(2)} \\ &= Z(A^{(1)},B^{(1)},\pi)+Z(A^{(2)},B^{(2)},\pi) \end{aligned} \tag{5.19}$$

Thus, the objective function of the BiQAP is the product of two QAP objective functions. Without loss of generality we only consider QAPs and BiQAPs with positive coefficients (and therefore positive objective function values).

We apply the QAP test generator of Li and Pardalos described in Section 2.6 to generate the $n\times n$ matrices $A^{(1)}$, $B^{(1)}$ and $A^{(2)}$, $B^{(2)}$. Recall that for the QAP test generator of Li and Pardalos, the optimal solution of the generated QAP instance is part of the input. Therefore, we may assume that the problems $QAP(A^{(i)},B^{(i)})$, $i=1,2$, have a common optimal solution, say π_0. If these matrices are then used to generate the coefficient arrays A and B of a BiQAP instance as in (5.18), the above common solution π_0 is also an optimal solution to BiQAP(A,B). This simple idea is summarized by the following algorithm. Notice that we have modified the algorithm proposed by Li and Pardalos in order to guarantee test instances with positive coefficients.

Algorithm 5.2 *Construction of a BiQAP instance with a given permutation as optimal solution.*
Input data: The optimal permutation π_0 given by its permutation matrix P.
Output data: Two arrays A and B of n^4 entries each containing the coefficients of the required BiQAP instance to which π_0 is an optimal solution.

Initialize *Input four integers* $\Delta_1 > 1$, $\Delta_2 > 1$, $\Delta_3 > 1$, $\Delta_4 > 1$.
Construct $A^{(1)}$ *and* $A^{(2)}$ *by setting* $a_{ij}^{(1)} = \Delta_1$ *and* $a_{ij}^{(2)} = \Delta_2$, $i, j = 1, 2, \ldots, n$, $i \neq j$. *The diagonal entries of both matrices are set to 0.*
Randomly generate integers $b_{ij}^{(1)}$ *and* $b_{ij}^{(2)}$, $i, j = 1, 2, \ldots, n$, *uniformly distributed on* $[1, \Delta_3]$ *and* $[1, \Delta_4]$, *respectively.*

Sort *Sort* $b_{ij}^{(1)}$ *and* $b_{ij}^{(2)}$, $i \neq j$, *increasingly.*
Store the corresponding ranks in $r_{ij}^{(1)}$ *and* $r_{ij}^{(2)}$, $i, j = 1, 2, \ldots, n$, $i \neq j$, *respectively, where for each b-coefficient its rank is the number which indicates its position in the respective ordering.*

Random *Randomly generate nonnegative integers* $x_i < \Delta_1$, $y_i < \Delta_2$, $i = 1, 2, \ldots, n(n-1)$, *and sort them increasingly.*

Update *Set* $a_{ij}^{(1)} := a_{i,j}^{(1)} - x_{r_{ij}^{(1)}}$ *and* $a_{ij}^{(2)} := a_{ij}^{(2)} - y_{r_{ij}^{(2)}}$, $i, j = 1, 2, \ldots, n$, $i \neq j$.

Permute *Set* $B^{(i)} := P^{-1} B^{(i)} P$, $i = 1, 2$, *where* P^{-1} *is the inverse of matrix* P.

Multiply *Generate the coefficients* a_{ijkl}, b_{ijkl}, $i, j, k, l = 1, 2, \ldots, n$, *of BiQAP(A,B) as described in (5.18). Output* $A = (a_{ijkl})$, $B = (b_{ijkl})$.

The correctness of this algorithm immediately follows from the correctness of Algorithm 2.2 which summarizes the generator of Li and Pardalos for QAPs with known optimal solution. The correctness of the latter algorithm is stated by Theorem 2.11 formulated and proven in the first part of this thesis.

The five first steps of Algorithm 5.2, which are also its essential steps, take $O(n^2 \log n)$ time (cf. [117]), whereas the last step *Multiply* obviously takes $O(n^4)$ time. Hence, the overall time complexity of this algorithm amounts to $O(n^4)$. Notice that this is the best possible time complexity for a BiQAPs generator, as the output size is already $O(n^4)$.

Next, let us apply Algorithm 5.2 for generating a BiQAP instance of size 3 with known optimal solution.

Example 5.1 *Generating a BiQAP instance with known optimal solution.*
Let $\Delta_1 = 3$, $\Delta_2 = 2$, $\Delta_3 = 3$, $\Delta_4 = 3$. Then the matrices $A^{(1)}$ and A^2 are given as

$$A^{(1)} = \begin{pmatrix} 0 & 3 & 3 \\ 3 & 0 & 3 \\ 3 & 3 & 0 \end{pmatrix} \qquad A^{(2)} = \begin{pmatrix} 0 & 2 & 2 \\ 2 & 0 & 2 \\ 2 & 2 & 0 \end{pmatrix}.$$

Let the randomly generated matrices $B^{(1)}$ and $B^{(2)}$ be given as follows:

$$B^{(1)} = \begin{pmatrix} 2 & 1 & 1 \\ 3 & 2 & 2 \\ 1 & 3 & 1 \end{pmatrix} \qquad B^{(2)} = \begin{pmatrix} 2 & 3 & 1 \\ 3 & 3 & 2 \\ 2 & 2 & 2 \end{pmatrix}.$$

Consider two vectors X and Y such that $X^t = (0,0,2,2,2,2)$ and $Y^t = (0,0,0,1,1,1)$. Then, the transformation of matrices $A^{(i)}$, $i = 1,2$, as described in Step *Update* yields:

$$A^{(1)} = \begin{pmatrix} 0 & 3 & 3 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix} \qquad A^{(2)} = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 0 & 2 \\ 2 & 1 & 0 \end{pmatrix}.$$

Next, consider a permutation $\pi_0 = \langle 3,1,2 \rangle$ and permute the rows and the columns of both matrices $B^{(1)}$ and $B^{(2)}$ by this permutation. We get

$$B^{(1)} = \begin{pmatrix} 2 & 2 & 3 \\ 3 & 1 & 1 \\ 1 & 1 & 2 \end{pmatrix} \qquad B^{(2)} = \begin{pmatrix} 3 & 2 & 3 \\ 2 & 2 & 2 \\ 3 & 1 & 2 \end{pmatrix}.$$

Finally, applying (5.18) with matrices $A^{(1)}, A^{(2)}$ and $B^{(1)}, B^{(2)}$ as above, yields a BiQAP instance of size 3, to which π_0 is an optimal solution. The corresponding optimal value is 1156.

We tested the lower bounds derived in Section 5.3 on test instances with known optimal solution generated by Algorithm 5.2. For the computation of the lower bounds two versions of Algorithm 5.1 were implemented, using either of the two reduction methods in Step 1. None of these methods was found to give consistently better results than the other. Then, we used the first reduction method in Step 1 of Algorithm 5.1 in all our further tests. Algorithm 5.2 was implemented as a MATLAB code in a DIGITAL DECstation 5000/240. Being conditioned by memory capacities, the size of BiQAP instances which can be generated in this way does not exceed 70, while the coefficients a_{ijkl} and b_{ijkl} are integers between 1 and 25. We tested the lower bounds on a benchmark consisting of BiQAP instances of size n, $n = 5, \ldots, 9$, with integer coefficients between 1 and 25. These test runs ended up with a huge relative error ranging between 20% and 35%. Taking into account that the coefficients of BiQAPs arising from practical applications as described in Section 5.1 are small, we made a second test run. The benchmark consisted again of BiQAP instance of size 5 up to 9, but the coefficients were integers between 1 and 9. The second test runs ended up with tighter but still unsatisfactory bounds, with corresponding relative error ranging between 7% and 22%.

It is easy to realize why these lower bounds are not good: they are based on linearizing a power-four function, a mathematical approach which produces a huge error even in very simple models. Due to the weakness of the lower bounds, branch and bound codes for BiQAPs will show a poor behavior, in particular when n increases. Therefore, it is an important task to design good heuristics for BiQAPs. In the next chapter we report on computational experiments with local search and simulated annealing type approaches for BiQAPs.

5.5 The asymptotic behavior of BiQAPs

In Section 2.3 it is shown that the approximation problem for QAPs is $\mathcal{NP}$-hard. Since QAPs are special cases of BiQAPs, as shown in Section 5.2, even the approximation problem for the latter is $\mathcal{NP}$-hard. Nevertheless, similarly as for QAPs, the approximation problem for BiQAPs becomes in a certain sense simple, when the problem size increases. Namely, it can be shown that under natural probabilistic constraints, the ratio between the best and the worst value of the BiQAP objective function gets arbitrarily close to one with probability tending to one, as the size of the problem tends to infinity. Similarly as in the case of QAPs, it is easy to apply Theorems 2.12 and 2.13 to BiQAPs, whose coefficients fulfill appropriate probabilistic constraints. Then, one would get for BiQAPs a corollary analogous to Corollary 2.14 for QAPs. Here, we apply Theorem 2.12 to get a slightly stronger result concerning the convergence of the about mentioned ratio in probability. First of all let us introduce the following useful notational conventions. For a BiQAP instance of size n, $\mathrm{BiQAP(A^{(n)},B^{(n)})}$, where $A^{(n)} = \left(a^{(n)}_{ijkl}\right)$ and $B^{(n)} = \left(b^{(n)}_{ijkl}\right)$, let us denote by $\pi^{(n)}_{opt}$ and $\pi^{(n)}_{wor}$ a best and a worst solution respectively, i.e.,

$$Z(A^{(n)},B^{(n)},\pi^{(n)}_{opt}) = \min_{\pi\in\mathcal{S}_n} Z(A^{(n)},B^{(n)},\pi)$$

$$Z(A^{(n)},B^{(n)},\pi^{(n)}_{wor}) = \max_{\pi\in\mathcal{S}_n} Z(A^{(n)},B^{(n)},\pi)$$

Corollary 5.3 *Consider a sequence of problems BiQAP($A^{(n)},B^{(n)}$), $n\in\mathbb{N}$, with coefficient arrays $A^{(n)} = \left(a^{(n)}_{ijkl}\right)$, $B^{(n)} = \left(b^{(n)}_{ijkl}\right)$ of n^4 elements each. Consider moreover the sets I_n, $n\in\mathbb{N}$, such that $I_n \subseteq \{1,2,\ldots,n\}^4$.*
Assume that $a^{(n)}_{ijkl}$, $n\in\mathbb{N}$, $(i,j,k,l)\in I_n$, are identically, independently distributed random variables on $[0,M]$, where M is some positive constant. Moreover, these variables have finite mean and variance. The other a-coefficients, $a^{(n)}_{ijkl}$, $(i,j,k,l)\notin I_n$, are all equal to zero.
Furthermore, assume that $b^{(n)}_{ijkl}$, $n\in\mathbb{N}$, $i,j,k,l = 1,2,\ldots,n$, are also identically, independently distributed random variables on $[0,M]$ with finite mean and variance. Finally, let the variables $a^{(n)}_{ijkl}$ and $b^{(n)}_{ijkl}$, $n\in\mathbb{N}$, $i,j,k,l = 1,2\ldots,n$, be pairwise independently distributed.
If, additionally, the following holds

$$\lim_{n\to\infty}\frac{|I_n|}{n\ln n} = \infty\,, \tag{5.20}$$

then

$$\forall\epsilon>0\,,\quad \lim_{n\to\infty} P\left\{\frac{Z(A^{(n)},B^{(n)},\pi^{(n)}_{wor})}{Z(A^{(n)},B^{(n)},\pi^{(n)}opt)} < 1+\epsilon\right\} = 1\ . \tag{5.21}$$

Proof. The proof is almost trivial. First, notice that Theorem 2.12 remains true when considering problems with coefficients on $[0, M]$ instead of $[0, 1]$. Next, let us remark that the above claim trivially holds, when either the expected value of the a-coefficients, denoted by $E(a)$, equals zero, or the variance of the b-coefficients, denoted by $\sigma^2(b)$, is equal to zero. In the first case, the coefficients a_{ijkl} would be equal to zero, for all $i, j, k, l = 1, 2, \ldots, n$. This would imply $Z(A^{(n)}, B^{(n)}, \pi) = 0$, for all $\pi \in \mathcal{S}_n$. In the second case, the b coefficients would be constant and therefore the objective functions $Z(A^{(n)}, B^{(n)}, \pi)$, $n \in \mathbb{N}$, would be constant on $\pi \in \mathcal{S}_n$. Through the rest of the proof we assume that $E(a) > 0$ and $\sigma^2(b) > 0$ and refer to the formulation of the BiQAP in the frame of the general definition of combinatorial optimization problems, given in Subsection 1.3.6.

In our case, the ground sets $\mathcal{E}_n$, $n \in \mathbb{N}$, the feasible solutions $X_n^{(\pi)}$ corresponding to $\pi \in \mathcal{S}_n$, and the sets of feasible solutions $\mathcal{F}_n$, $n \in \mathbb{N}$, are given as follows:

$$\begin{aligned}
\mathcal{E}_n &= \Big\{ (i, j, k, l, m, p, s, t) \in \{1, 2, \ldots, n\}^8 \, : \, (i, j, k, l) \in I_n \Big\} \\
X_n^{(\pi)} &= \Big\{ (\pi(i)\pi(j)\pi(k)\pi(l), i, j, k, l) \colon (\pi(i), \pi(j), \pi(k), \pi(l)) \in I_n \Big\} \\
\mathcal{F}_n &= \Big\{ X_n^{(\pi)} \colon \pi \in \mathcal{S}_n \Big\}.
\end{aligned}$$

Notice that the feasible solution $X_n^{(\pi)}$ consists only of those 8-tuples $(\pi(i), \pi(j), \pi(k), \pi(l), i, j, k, l)$ for which the corresponding coefficient $a_{\pi(i)\pi(j)\pi(k)\pi(l)}$ does not equal zero.

Since some of the sets $X_n^{(\pi)}$, $\pi \in \mathcal{S}_n$, might be identical, we have $1 \leq |\mathcal{F}_n| \leq n!$. Moreover, we clearly have $|X_n^{(\pi)}| = |I_n|$, for each $\pi \in \mathcal{S}_n$ and any $n \in \mathbb{N}$. Thus, under the conditions of the theorem, the following holds for each $\lambda_0 > 0$:

$$\lambda_0 |X_n^{(\pi)}| - \ln(|\mathcal{F}_n|) \geq \lambda_0 |I_n| - \ln(n!) \to \infty \quad \text{as} \quad n \to \infty.$$

Next, we show that the variables $f_n(x)$, $x \in \mathcal{E}_n$, have positive variance, where f_n is the cost function related to $\mathrm{BiQAP}(\mathrm{A}^{(\mathrm{n})}, \mathrm{B}^{(\mathrm{n})})$. Recall that

$$f_n(x) = a^{(n)}_{\pi(i)\pi(j)\pi(k)\pi(l)} b^{(n)}_{ijkl} \, ,$$

where $x = (\pi(i), \pi(j), \pi(k), \pi(l), i, j, k, l)$, $(\pi(i), \pi(j), \pi(k), \pi(l)) \in I_n$.

Notice that for independently distributed random variables a and b we have:

$$\sigma^2(ab) = E((ab - E(ab))^2) = E(a^2)\sigma^2(b) + E(b^2)\sigma^2(a)$$

Now, the claim follows when plugging in the above equality $a^{(n)}_{\pi(i)\pi(j)\pi(k)\pi(l)}$ and $b^{(n)}_{ijkl}$ instead of a and b respectively, while taking into account that $E(a^2) > 0$ and $\sigma^2(b) > 0$.

The other conditions of Theorem 2.12 are trivially fulfilled. Hence, from this theorem follows (5.21). ■

Theorem 2.12 applies even if some, but not too many, of the coefficients $b_{ijkl}^{(n)}$ are fixed to zero. More precisely, we get the following straightforward corollary:

Corollary 5.4 *Consider the same sequence of problems $BiQAP(A^{(n)}, B^{(n)})$ as in the previous corollary and two subsets I_n, J_n of $\{1, 2, \ldots, n\}^4$.*
Let $a_{ijkl}^{(n)}$, $n \in \mathbb{N}$, $(i, j, k, l) \in I_n$, be identically, independently distributed random variables on $[0, M]$, for some positive constant M. Let they have finite mean and variance.
Analogously, $b_{ijkl}^{(n)}$, $n \in \mathbb{N}$, $i, j, k, l \in J_n$, are identically, independently distributed random variables on $[0, M]$, with positive variance $\sigma^2(b)$ and finite mean. Moreover, the variables $a_{ijkl}^{(n)}$ and $b_{mpst}^{(n)}$, $n \in \mathbb{N}$, $i, j, k, l, m, p, s, t = 1, 2, \ldots, n$, are pairwise independent.
Finally, all the other coefficients, $a_{ijkl}^{(n)}$, $(i, j, k, l) \notin I_n$, $b_{mpst}^{(n)}$, $(m, p, s, t) \notin J_n$, are equal to zero.
Let us denote

$$\alpha_{ijkl} = \begin{cases} 1 & (i, j, k, l) \in I_n \\ 0 & otherwise \end{cases} \qquad \beta_{mpst} = \begin{cases} 1 & (m, p, s, t) \in J_n \\ 0 & otherwise \end{cases}$$

and

$$c(n) = \min_{\pi \in \mathcal{S}_n} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} \alpha_{\pi(i)\pi(j)\pi(k)\pi(l)} \beta_{ijkl}.$$

If, additionally, the following holds

$$\lim_{n \to \infty} \frac{c(n)}{n \ln n} = \infty\,, \tag{5.22}$$

then

$$\forall \epsilon > 0\,, \qquad P\left\{ \frac{Z(A^{(n)}, B^{(n)}, \pi_{wor}^{(n)})}{Z(A^{(n)}, B^{(n)}, \pi_{opt}^{(n)})} < 1 + \epsilon \right\} \to 1\,. \tag{5.23}$$

Proof. In this case, the ground sets $\mathcal{E}_n$, $n \in \mathbb{N}$, are given as

$$\mathcal{E}_n \quad = \quad \Big\{ (i, j, k, l, m, p, s, t) \colon (i, j, k, l) \in I_n,\ (m, p, s, t) \in J_n \Big\}.$$

In order to apply Theorem 2.12 we have to show that under the above assumptions the condition (2.23) of Theorem 2.12 is still fulfilled. But this is an immediate consequence of the equality $|X_n^*| = c(n)$ and the condition (5.22). ■

In Corollary 5.4, the condition (5.22) as well as the assumption $\sigma^2(b) > 0$ are essential, as will be shown by the following two examples.

Example 5.2 *The assumption $\sigma^2(b) > 0$ in Corollary 5.4 is essential.*
Consider BiQAP($\mathrm{A^{(n)}}, \mathrm{B^{(n)}}$), $A^{(n)} = \left(a_{ijkl}^{(n)}\right)$, $B^{(n)} = \left(b_{ijkl}^{(n)}\right)$, $i, j, k, l = 1, 2 \ldots, n$, where

$$a_{ijkl}^{(n)} = \begin{cases} 1 & l = 1 \\ 0 & otherwise \end{cases} \qquad b_{ijkl}^{(n)} = \begin{cases} 0 & l = 1,\ k \neq 1 \\ 1 & otherwise \end{cases} \tag{5.24}$$

In this case any feasible solution yields an objective function value equal to its cardinality. The number of nonzero a-coefficients is n^3 and the number of b-coefficients which are equal to zero is $(n-1)n^2$. Therefore, for any permutation (feasible solution) at most $(n-1)n^2$ nonzero a-coefficients will be mapped into b-coefficients equal to zero. Thus any feasible solution has a value $\geq n^2$, i.e., $c(n) \geq n^2$ and therefore (5.22) is fulfilled. The other conditions of Corollary 5.4 are trivially fulfilled, except for inequality $\sigma^2(b) > 0$, which is clearly violated.

We show that the claimed result does not hold. Indeed, let us consider two permutations π_1 and π_2, such that $\pi_1(1) = 1$ and $\pi_2(1) \neq 1$. It is easy to check that

$$Z(A^{(n)}, B^{(n)}, \pi_1) = n^2 \text{ and } Z(A^{(n)}, B^{(n)}, \pi_2) = n^3$$

$$\text{Thus,} \quad \lim_{n\to\infty} \frac{Z(A^{(n)}, B^{(n)}, \pi_{wor}^{(n)})}{Z(A^{(n)}, B^{(n)}, \pi_{opt}^{(n)})} \geq \lim_{n\to\infty} \frac{Z(A^{(n)}, B^{(n)}, \pi_2)}{Z(A^{(n)}, B^{(n)}, \pi_1)} = \infty\,.$$

Example 5.3 *The condition 5.22 in Corollary 5.4 is essential.*
Similarly as in the previous example, consider BiQAP($\mathrm{A^{(n)}}, \mathrm{B^{(n)}}$), whose coefficients are given as

$$a_{ijkl}^{(n)} = \begin{cases} 1 & l = 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.25}$$

$$b_{ijkl}^{(n)} = \begin{cases} 0 & l = 1 \\ \text{uniformly distributed in } [1/2,1] & l \neq 1 \end{cases} \tag{5.26}$$

Let π_1, π_2 be specified as in the previous example. It can easily be checked that in this case we have

$$Z(A^{(n)}, B^{(n)}, \pi_1) = 0 \text{ and } Z(A^{(n)}, B^{(n)}, \pi_2) \geq 1/2n^3$$

Again, all conditions of Corollary 5.4 are fulfilled, except of condition (5.22) which is clearly violated. The result claimed by the corollary does not hold, since

$$\lim_{n\to\infty} \frac{Z(A^{(n)}, B^{(n)}, \pi_{opt}^{(n)})}{Z(A^{(n)}, B^{(n)}, \pi_{wor}^{(n)})} \leq \lim_{n\to\infty} \frac{Z(A^{(n)}, B^{(n)}, \pi_1)}{Z(A^{(n)}, B^{(n)}, \pi_2)} = 0\,.$$

Chapter 6

Heuristic Approaches for the BiQAP and Their Computational Comparison

The relationship between QAPs and BiQAPs described in the previous chapter implies that the BiQAP is at least as hard as the QAP, from both the theoretical and the practical point of view. Moreover, the quality of Gilmore-Lawler type bounds introduced in Section 5.3 is unsatisfactory, especially when the size of the problem increases. Therefore, it is quite unlikely that branch and bound algorithms involving these bounds show a good behavior. Under these conditions, heuristic approaches are a reasonable effort to cope with BiQAPs. Clearly, the aim is to derive such heuristics which yield relatively good suboptimal solutions and come about with acceptable requirements of computational resources (e.g. memory and time requirements).

In this chapter we introduce several heuristics for BiQAPs, in particular pair exchange algorithms (deterministic improvement methods) and variants of simulated annealing and tabu search (stochastic improvement methods). We implement these heuristics as C codes and analyze their performances. Our comparative study focuses on a trade off between solution quality and computation time. The comparison is done on benchmarks consisting of BiQAP instances of size 10 up to 32 with known optimal solution. These instances are produced by the generator described in Section 5.4.

The rest of this chapter is organized as follows. In the next section, the deterministic improvement methods used in this experimental study are discussed. In this context and in analogy with QAPs, it is shown that the neighborhood evaluation can be performed in $O(n^4)$ time, where n is the size of the problems. This clearly is the lowest possible time cost, as in the case of BiQAPs the evaluation of the objective function itself amounts to $O(n^4)$ elementary operations. Then, in the second section we describe the so called *stochastic improvement methods* for BiQAPs. In the first part of this section we introduce three variants of simulated annealing for BiQAPs. One of them is a generalization of a classical the simulated annealing approach for

QAPs proposed by Wilhelm and Ward [184]. The second variant is a modification of the first one by dropping the equilibrium criterion, whereas the third version is based on the notion of the so called *optimal temperature*, originally introduced under this name by Connolly [45] in 1990. Further on, in the second part of this section, we describe a tabu search approach for BiQAPs and also introduce a hybrid version of tabu search and simulated annealing. In the third section, a detailed analysis and comparison of all proposed approaches is presented. Taking into account the decisive role of the control parameters in the performance of the corresponding (stochastic) improving approaches, we try to find out the optimal range of different control parameters. Moreover, some comparative results concerning the impact of different control parameters on the quality of heuristics performance are presented. This point of view becomes interesting when working with algorithms which involve a high number of parameters to be tuned. (The parameter tuning usually takes a lot of efforts and turns out to be the main drawback of heuristic approaches such as simulated annealing and tabu search.)

Finally, we summarize the results obtained on BiQAPs and outline some open questions for further research in a concluding section.

6.1 Deterministic improvement methods

In Subsection 2.5, we gave a general description of deterministic improvement methods and tabu search approaches for QAPs. Analogously as for QAPs, we consider the following three approaches for BiQAPs:

1. First improvement method.
2. Best improvement method.
3. Heider's improvement method.

From now on the "first improvement method", the "best improvement method" and the "Heider's improvement method" will be abbreviated by BEST, FIRST, HEID, respectively. Without repeating the general description of BEST, FIRST and HEID, we only notice that the so called *pair-exchange* neighborhood (introduced in Subsection 2.5) is exclusively used throughout this chapter. More precisely, let the set of transpositions on $\{1, 2, \ldots, n\}$ be denoted by I. Then, for an arbitrary permutation $\pi_0 \in \mathcal{S}_n$, its neighborhood $\mathcal{N}(\pi_0)$ on $\mathcal{S}_n$ is given by

$$\mathcal{N}(\pi_0) = \Big\{(i, j) \circ \pi_0 \colon (i, j) \in I\Big\} \tag{6.1}$$

Let us notice that a fixed numbering of the elements of I is considered throughout the rest of this chapter. Moreover, the so called *Stop criterion* involved in the above three improvement methods is the standard one. Namely, when dealing with BiQAP(A,B) the stop criterion looks as follows:

If $Z(A, B, \pi) \geq Z(A, B, \pi_0)$ for each $\pi \in \mathcal{N}(\pi_0)$, where π_0 is the current permutation, then stop.

Next, let us analyze the computational effort involved when implementing FIRST, BEST and HEID. These methods show the same worst case behavior. Assume we are applying FIRST, BEST or HEID to BiQAP(A,B) of size n and $\pi_0 \in \mathcal{S}_n$ is our current solution. Clearly, in the worst case $n(n-1)/2$ differences $\Delta Z = Z(A, B, \pi_0) - Z(A, B, \pi)$, $\pi \in \mathcal{N}(\pi_0)$, must be computed. Let us pick up a permutation $\pi = (k_0, l_0) \circ \pi_0$, $(k_0, l_0) \in I$, in the neighborhood of π_0. The corresponding ΔZ can be trivially computed in $O(n^4)$ time, since computing each of $Z(A, B, \pi)$ and $Z(A, B, \pi_0)$ involves $O(n^4)$ elementary operations. A less straightforward computation of ΔZ can be performed in $O(n^3)$ time. This becomes clear when considering the following equalities and notational conventions:

$$K = \Big\{(i, j, k, l) \ : \ i, j, k, l \in \{1, 2, \ldots, n\}\Big\} ,$$

$$K_1 = \Big\{(i, j, k, l) \in K \colon \{i, j, k, l\} \cap \{k_0, l_0\} \neq \emptyset\Big\}, \quad K_2 = K \setminus K_1 .$$

Obviously, $(i, j, k, l) \in K_2$ implies $a_{\pi_0(i)\pi_0(j)\pi_0(k)\pi_0(l)} \cdot b_{ijkl} = a_{\pi(i)\pi(j)\pi(k)\pi(l)} \cdot b_{ijkl}$. Considering this fact, we get

$$\Delta Z = \sum_{(i,j,k,l)\in K_1} \Big[a_{\pi_0(i)\pi_0(j)\pi_0(k)\pi_0(l)} - a_{\pi(i)\pi(j)\pi(k)\pi(l)}\Big] \cdot b_{ijkl} \tag{6.2}$$

Now, it can easily be seen that $|K_1| = |K| - |K_2| = n^4 - (n-2)^4 = O(n^3)$. Hence, the sum in (6.2), which has $|K_1|$ terms, can be computed in $O(n^3)$ time.

What about the number of iterations any improvement method runs through? A trivial theoretical upper bound is $n!$, whereas in our computational experiments the number of iterations never exceeded $5n$.
As the neighborhood size equals $n(n-1)/2$, in each iteration of BEST $n(n-1)/2$ differences similar to ΔZ are computed. Then, each iteration would take $O(n^5)$ time, when computing ΔZ as described above. For HEID and FIRST the practical number of ΔZ calculations is not so high, although the theoretical bound is still $n(n-1)/2$. In the following it is shown that BEST can be speeded up.

For a current permutation π_0, consider all $(i, j) \in I$ and compute the respective ΔZ as in formula (6.2). Let us denote

$$\Delta Z\,(\pi_0, i, j) := Z(A, B, \pi_0) - Z(A, B, (i, j) \circ \pi_0) .$$

Then we save all differences $\Delta Z(\pi_0, i, j)$, $(i, j) \in \mathcal{S}_n$, as nondiagonal entries of a symmetric $n \times n$ matrix $D = \{d_{ij}\}$ with zeros on the diagonal.

$$d_{ji} = d_{ij} = \Delta Z(\pi_0, i, j) , \quad (i, j) \in I .$$

Let us suppose that at the end of the current iteration $\pi_1 = (i_0, j_0) \circ \pi_0$ becomes the current permutation. In the next iteration we must compute

$$d'_{ij} = d'_{ji} = \Delta Z(\pi_1, i, j) = Z(A, B, \pi_1) - Z(A, B, (i,j) \circ \pi_1)\,, \quad (i,j) \in I\,,$$

where $D' = (d'_{ij})$ is an $n \times n$ matrix with zero diagonal entries. The entries of matrix D' can be computed in $O(n^4)$ time, as shown below. Consider $(k_0, l_0) \in I$, such that $\{k_0, l_0\} \cap \{i_0, j_0\} = \emptyset$, $\pi_2 = (k_0, l_0) \circ \pi_1$, and let us compute $d'(k_0 l_0) = \Delta Z(\pi_1, k_0, l_0)$. First, we introduce two more notations:

$$Q_1 = \{(i,j,k,l)\colon (i,j,k,l) \in K_1,\ \{i,j,k,l\} \cap \{i_0, j_0\} = \emptyset\}\,, \quad Q_2 = K_1 \setminus Q_1$$

Considering (6.2) we get:

$$\begin{aligned} d'_{k_0 l_0} &= \sum_{(i,j,k,l) \in Q_1} \left[a_{\pi_1(i)\pi_1(j)\pi_1(k)\pi_1(l)} - a_{\pi_2(i)\pi_2(j)\pi_2(k)\pi_2(l)}\right] \cdot b_{ijkl} \\ &+ \sum_{(i,j,k,l) \in Q_2} \left[a_{\pi_1(i)\pi_1(j)\pi_1(k)\pi_1(l)} - a_{\pi_2(i)\pi_2(j)\pi_2(k)\pi_2(l)}\right] \cdot b_{ijkl} \end{aligned} \tag{6.3}$$

Let $\pi'_2 = (k_0, l_0) \circ \pi_0$. For $i \notin \{i_0, j_0\}$, we clearly have $\pi_1(i) = \pi_0(i)$ and $\pi_2(i) = \pi'_2(i)$. Taking into account this last equalities we get from (6.3)

$$\begin{aligned} d'_{k_0 l_0} &= d_{k_0 l_0} + \sum_{(i,j,k,l) \in Q_2} \left[a_{\pi_1(i)\pi_1(j)\pi_1(k)\pi_1(l)} - a_{\pi_0(i)\pi_0(j)\pi_0(k)\pi_0(l)}\right] \cdot b_{ijkl} \\ &+ \sum_{(i,j,k,l) \in Q_2} \left[a_{\pi'_2(i)\pi'_2(j)\pi'_2(k)\pi'_2(l)} - a_{\pi_2(i)\pi_2(j)\pi_2(k)\pi_2(l)}\right] \cdot b_{ijkl} \end{aligned} \tag{6.4}$$

Let us now compute $|Q_2|$. For any $(i,j,k,l) \in Q_2$, we have $\{i,j,k,l\} \cap \{i_0, j_0\} \neq \emptyset$ and $\{i,j,k,l\} \cap \{k_0, l_0\} \neq \emptyset$. Therefore, one component of each 4-tuple $(i,j,k,l) \in Q_2$ takes its value on $\{i_0, j_0\}$ and another one takes its value on $\{k_0, l_0\}$ (recall that $\{i_0, j_0\} \cap \{k_0, l_0\} = \emptyset$). It is easy to see that these two facts imply $|Q_2| = 8 \cdot C_4^2 \cdot n^2$. Hence, d'_{kl} can be computed in $O(n^2)$ time, for any (k,l) such that $\{k,l\} \cap \{i_0, j_0\} = \emptyset$. For the other pairs (k,l), that is for pairs (k,l) with $\{k,l\} \cap \{i_0, j_0\} \neq \emptyset$, d'_{kl} is computed as in (6.2). Since there are $4n - 7$ such pairs (k,l), the corresponding entries of D' can be computed in $O(n^4)$ time. Summarizing, we have proven the following theorem:

Theorem 6.1 *The "best improvement method"* BEST *for BiQAPs can be implemented such that each iteration, except of the first one, takes $O(n^4)$ time, where n is the size of the BiQAP instance.* ■

6.2 Stochastic improvement methods

This section is devoted to so called stochastic improvement methods for BiQAPs. In analogy with deterministic improvement methods, the stochastic approaches aim also to improve the objective function value in each iteration by focusing on the so called *down-hill* moves. The difference concerns so called *up-hill* moves, which in the case of deterministic methods are excluded (with probability equal to one), whereas here such moves are accepted with a positive but small probability. The stochastic approaches show a better performance with respect to the solution quality, as the stochastic component cares for a diversification of the search on a larger area of the feasible region.

6.2.1 Three variants of simulated annealing

The simulated annealing scheme, which is successfully applied to several hard optimization problems, can be applied to BiQAPs too. We have implemented three variants of simulated annealing for BiQAPs. The first one is a generalization of simulated annealing for QAPs implemented by Wilhelm and Ward [184]. The cooling process drops from the current temperature value to the next one in the schedule iff, either the equilibrium is reached or enough neighboring permutations are considered at the current temperature value. Therefore, at each temperature value the amount of the considered neighboring permutations does not exceed a certain number, say Max, which is controlled by one of the control parameters. This control parameter is denoted by N_2. Since ΔZ is calculated as in (6.2), each cooling step runs in $Max \cdot O(n^3)$ time and the whole method takes $r \cdot Max \cdot O(n^3)$ time, where r is the amount of cooling steps. If $Max = n \cdot N_2$, as in all our experiments, the whole method takes $r \cdot N_2 \cdot O(n^4)$ time. We usually select N_2 and r such that $N_2 \leq 10$ and $r = O(n)$ and the time complexity amounts to $O(n^5)$. The temperature schedule involved in this version of simulated annealing is defined by $t_k = q^k t_0$, where t_k is the k-th temperature value in the schedule, $0 \leq k \leq r$, and $q \in (0, 1)$ is another control parameter. A detailed discussion on strategies for setting the control parameter values follows in Section 6.3.

We propose a second version of simulated annealing for BiQAPs. Its new features concern three main points:

1. The equilibrium criterion has been eliminated. The amount of attempted transpositions at a given temperature value does not exceed $K \cdot n(n-1)$, where K is a control parameter. The temperature drops from the current state to the next one in the schedule iff, either a solution is found which yields a decrease in the objective function value and hence is accepted, or $K \cdot n(n-1)$ transpositions have already been attempted.

2. A numbering in I is fixed and the transposition to be applied to the current

permutation is not randomly chosen, but is the next transposition in this numbering. Once a transposition is accepted, the current permutation is updated and the transpositions of I are again considered starting from the first one in the fixed order.

3. Another formula is used for generating the temperature schedule, namely

$$t_{k+1} := \frac{t_k}{1+\beta \cdot t_k}, \tag{6.5}$$

where β is a control parameter, $0 < \beta \ll t_0$[1].

Similarly as above, a time complexity analysis shows that the method takes $r \cdot K \cdot O(n^5)$ time. (From now on r is the amount of cooling steps unless otherwise specified.) If $r = O(n)$, this method takes $O(n^6)$ time.

Computational results show that the range of the temperature schedule is very important for the performance of simulated annealing approaches. If the system is kept too "hot", neighboring permutations which yield a big increase of the objective function value are accepted. Therefore, it may be impossible to get to a local minimum. If the system is kept too "cold", only neighboring permutations which yield very small increases or decreases of the objective function value are accepted. In this case the search may easily get stuck in a bad local minimum. This leads to the idea that somewhere between these two extremes there must be an *optimal temperature value*. Several efforts are made to elaborate an appropriate concept of the optimal temperature value when applying simulated annealing to different problems (see [104] and [181]). In [45] several manners to find the optimal temperature in case of QAPs are proposed and respective computational results are reported. Computational results on QAPs showed that the more a standard annealing search was performed at or close to the optimal temperature, the more successful the search became. It is reasonable to expect a similar behavior of BiQAPs.
In the following, a method for deriving the optimal temperature value for BiQAPs out of a given temperature schedule is proposed. First, the optimal temperature value should not be too high, in order to avoid a relatively high percentage of accepted transpositions for a fixed amount of attempted ones. Such high acceptance rates would lead to a chaotic search. Secondly, the optimal temperature value should not be too low, in order to avoid a too low percentage of accepted transpositions. Getting stuck in bad local minima would be a frequent consequence of such a low acceptance rate. Assuming that our temperature schedule extends to a large enough interval of temperature values, we may suppose that the acceptance rates are high and low for the first and the last temperatures in the schedule, respectively. Intuitively, we would

[1]Theoretical and empirical results suggest that a reasonably slow cooling process is necessary for a good performance of simulated annealing. The parameter β is chosen much smaller than t_0 just to ensure such a slow cooling process.

like most of the search to be performed at a temperature value whose acceptance rate approaches the average acceptance rate, for the temperature schedule at hand and the considered problem instance. Thus, in order to find such a temperature, we consider a fixed amount of transpositions and compute the respective percentage of accepted transpositions at each step in the cooling process. Then, the arithmetical mean of these percentages is computed and the deviations of all percentages from this mean are evaluated. The temperature value with the minimum corresponding deviation is chosen as the optimal one. Afterwards, an appropriate number of search iterations are performed at the optimal temperature value. It is worthy to notice that the cooling must be very slow, in order to get to a temperature value close enough to the optimal one.
These ideas lead to the third version of simulated annealing for BiQAPs which is described by the following algorithm. The algorithm runs through two phases. In the first one the optimal temperature value is derived, whereas in the second phase the search performs at the optimal temperature. K_1 and K_2 are control parameters as described below:

$\boldsymbol{K_1}$: control parameter related to the amount of transpositions attempted per cooling step, in the first phase of the algorithm. This amount is equal to $K_1 \cdot n(n-1)$.

$\boldsymbol{K_2}$: control parameter related to the amount of transpositions attempted in the second phase of the algorithm. This amount is equal to $K_2 \cdot n(n-1)$.

Algorithm 6.1 *Simulated annealing for BiQAPs, third version.*
Input data: The coefficients a_{ijkl} and b_{ijkl}, $i, j, k, l = 1, 2, \ldots, n$, of a problem instance BiQAP(A,B) of size n, the control parameters r, K_1, K_2, β and the initial value t_0 of the temperature.
Output: A suboptimal solution to BiQAP(A,B) and its corresponding objective function value.

First Phase

Initialize *Generate an initial permutation $\pi_0 \in \mathcal{S}_n$ (for example, randomly). Set $t := t_0$. Repeat the steps* Search, Accept and Update *$K_1 \cdot n(n-1)$ times.*

Search *Let (i, j) be the current transposition with respect to the fixed cyclic order in I. Set $\pi := (i, j) \circ \pi_0$. Goto* Accept and Update.

Accept and Update

Check *If one of the two following conditions is fulfilled then goto* Update.

1. $\Delta = Z(A, B, \pi) - Z(A, B, \pi_0) < 0$

2. $x < \exp(-\Delta/t)$, *where t is the current temperature and $x \sim U(0,1)$ is randomly taken from a uniform distribution on $(0,1)$.*

Update *Update the current solution $\pi_0 := \pi$ and the counter of the accepted transpositions. Update the best solution found so far and the corresponding objective function value.*

Percentage *Compute the percentage of the accepted transpositions with respect to the attempted ones. Goto* Cool down.

Cool down *Unless the final temperature is reached, decrease the temperature to the next value in the schedule: $t := \frac{t}{1+\beta t}$*

Optimal Temperature *Compute the arithmetical mean m of percentages computed by* Percentage *at each cooling step. Find a temperature value in the schedule with minimum deviation of the corresponding percentage from m. Save it as optimal temperature value t^*.*

Second Phase

Reinitialize *Generate an initial permutation $\pi_0 \in \mathcal{S}_n$ (for example randomly). Repeat $K_2 \cdot n(n-1)$ times the steps* Search *and* Accept and Update *of the first phase with $t := t^*$ and without counting the accepted transpositions.*

Output *Output the best solution found so far and the corresponding objective function value.*

This algorithm runs in $(K_1 \cdot r + K_2) \cdot O(n^5)$ time. If, as usually, $r = O(n)$, then the third version of simulated annealing takes $O(n^6)$ time. As we will see in Section 6.3, the performance of the method strongly depends on the control parameters t_0 and β. β must be chosen small enough in order to realize a better approximation of the optimal temperature.

6.2.2 Tabu search and its combination with simulated annealing

A description of tabu search techniques in general and tabu search schemes for QAPs in particular is given in Section 2.5. We introduce here a tabu search scheme for BiQAPs. Moreover, trying to relax the aspiration criterion of this algorithm, a combined approach of tabu search and simulated annealing is proposed.
The movement in the neighborhood of the current permutation is realized by transpositions applied to it. The tabu list is a FIFO list which consists of transpositions that correspond to penalized moves. A penalized move is accepted only if it yields an improvement of the best objective function value known so far. Thus, in this case the so called *aspiration condition* is the improvement of the currently best objective function value. Our implementation of tabu search may run through two phases.

At the first phase we begin with an empty tabu list, i.e., with a tabu list whose length is 0. At each iteration all non-tabu transpositions and those tabu transpositions which fulfill the aspiration condition are considered in the order established in I. Among these transpositions the one which yields the smallest objective function value is accepted. Then, this transposition is applied to the current solution and the latter is hence updated. The tabu list is updated by adding or moving the last accepted transposition to the end of the list, if it was a non-tabu or a tabu transposition, respectively. If the length of the resulting list exceeds the tabu length, the first tabu transposition in the list becomes a non-tabu one. After having repeated this procedure a fixed number of times, the first phase is over. Since the performance of tabu search depends on the initial permutation and on the tabu length, its first phase is repeated with another value of tabu length and/or another initial permutation. It may happen that other accepted transpositions at the beginning of the first phase lead to a better suboptimal solution at the end of it. Therefore, it may be reasonable to penalize a number of transpositions among those accepted at the beginning of the first phase. This is realized by starting a second phase with full tabu list, namely with the first full tabu list generated in the first phase. In principal, tabu search can proceed further on with a third phase, penalizing the "first directions" of search involved in the two first phases. However, computational experiments show an explosion of execution times for tabu search implementations with more than two phases.
The control parameters of tabu search for BiQAPs are the number of iterations K at each phase of the algorithm and the length of the tabu list. A simple analysis shows that each phase takes $K \cdot O(n^4)$ time, where K is the corresponding number of iterations.

The aim of tabu search is to escape from bad local minima. But, if only few tabu moves fulfill the aspiration condition, it may happen that this escape is too "fast", in the sense that the method does not dwell enough in the neighborhood of the current permutation. Therefore, good solutions get lost although, for example, they may be obtained by applying one of the tabu transposition. Hence, it seems reasonable to relax the aspiration condition such that more tabu moves fulfill it. Namely, some tabu moves which lead to a small increase of the objective function value are additionally taken into account. To realize this, probabilities $P(\pi) = \min\{\exp(-\Delta Z/t), 1\}$, $\pi \in \mathcal{S}_n$, are introduced, where $\Delta Z = Z(A,B,\pi) - Z(A,B,\pi_0)$, π_0 is the the best solution known so far and t is a simulated temperature value. In our model, permutation π improves the best known objective function value with probability $P(\pi)$. Then, a tabu move $(i,j) \in I$ is said to fulfill the aspiration condition if and only if $P(\pi) > x$, where $\pi = (i,j) \circ \pi_0$ and $x \sim U(0,1)$ is randomly taken form a uniform distribution on $(0,1)$.
The value t of the simulated temperature is the optimal value out of a prespecified temperature schedule, derived as described in the previous section. Thus, two phases of the algorithm can be distinguished. In the first phase, *the temperature phase*, the optimal temperature is derived and in the the second phase, *the tabu phase*, a tabu

search with relaxed aspiration condition is applied.
This version of tabu search involves two additional control parameters: the temperature schedule and the iteration number K per cooling step, at the temperature phase. The temperature schedule is generated by $t_i := t_0 \cdot q^{i-1}$, where the coefficient $q \in (0,1)$, the initial temperature t_0 and the length r of the schedule are control parameters. If $K = O(n)$, the running time of the method is dominated by the running time of the temperature phase. Summarizing, this version of tabu search performs in $O(n^6)$ time, if r is set according to the usual choice $r = O(n)$.

6.3 Computational results

In this section we report on our computational experience with the heuristic approaches described in the previous sections. The tests are performed on BiQAP instances with known optimal solution generated by the algorithm described in Section 5.4, implemented as a MATLAB code in a DIGITAL DECstation 5000/240. The sizes of our test instances are even integers on $[10, 40]$ and the integer coefficients a_{ijkl}, b_{ijkl} range between 1 and 9. The test problem of size n is denoted P_n.

The initialization of all considered heuristics involves the generation of an initial permutation. 40 initial permutations are generated randomly for each test instance. The i-th initial permutation for the test instance of size n is denoted $snpi$, for example, $s18p3$ is the third initial permutation for the test instance P_{18}.

The heuristics and the generation of initial permutations are implemented as C codes in the above mentioned machine. These codes are called HEID, FIRST and BEST for the Heider's, first and best improvement methods, ANNEAL, SIMANN2 and SIMANN3 for the first, second and third version of simulated annealing, and TAB, LTMTAB and TABSIM for one phase tabu search, more phases tabu search and tabu search combined with simulated annealing, respectively.

The numerical results show the clear priority of HEID in comparison with FIRST and BEST. However, its performance becomes infeasible as the size of the problem increases. In our experiments, the average number of considered transpositions at a fixed iteration of FIRST and HEID does not exceed $5n$. In Table 6.1 the average relative error, the average CPU time, the iteration number and the frequency of finding an optimal solution are reported for HEID, FIRST and BEST.

The increase of the problem size over 36 is considered as fruitless since already for sizes larger than 30 the CPU time required by each improvement code exceeds 1 hour! In Figure 6.1 a graphical comparison of improvement codes is presented. The CPU times and the relative errors of HEID, FIRST and BEST are compared. Figure 6.1 shows that among the three considered improvement methods, HEID is the best trade off between the solution quality and CPU time requirements.

In the case of one phase tabu search TAB, there are two control parameters, namely, the tabu length and the iteration number. If the number of iterations exceeds

Table 6.1: Comparison of deterministic improvement methods

Size	Code	Rel. Error (in %)	CPU (in sec.)	It. Nr.	Opt. sol. (in %)
10	FIRST	6.80	6.19	10	70.2
	BEST	6.23	8.03	6	71.6
	HEID	6.08	4.14	12	73.7
14	FIRST	7.48	47.70	17	26.8
	BEST	9.76	54.05	8	10.3
	HEID	9.49	23.64	17	7.8
16	FIRST	11.04	62.62	18	4.6
	BEST	8.00	120.92	10	24.7
	HEID	5.25	65.20	20	55.9
20	FIRST	17.87	306.34	26	57.8
	BEST	22.60	456.16	14	5.3
	HEID	11.53	212.35	31	48.2
24	FIRST	6.68	870.68	29	3.7
	BEST	6.51	1213.98	16	4.6
	HEID	6.98	520.88	26	2.9
28	FIRST	9.02	3250.57	52	48.9
	BEST	12.93	3271.46	42	23.1
	HEID	4.49	1325.20	48	76.8
32	FIRST	18.47	6304.34	63	5.7
	BEST	17.26	6328.24	18	6.8
	HEID	18.52	4458.86	43	24.5
36	FIRST	20.02	9722.81	71	3.2
	BEST	18.90	9275.10	25	4.3
	HEID	13.84	4087.96	43	6.7

30, the CPU time required, even for BiQAP instances of size 20, grows so much that the practical use of the code becomes impossible. As the tabu length value does not need to exceed the iteration number, the latter remains the only relevant control parameter for TAB. Indeed, numerical results show that the performance of TAB applied to P_n with a fixed number of iterations depends very slightly on the values of the tabu length once these values exceed $2n$.

We applied TAB to a BiQAP instance of size 20 with different values for the iteration number. It turned out that the increase of the number of iterations causes a drastic decrease of the relative error, but at high costs of CPU time. Table 6.2 summarizes results of test runnings with 40 iterations each and tabu list of length

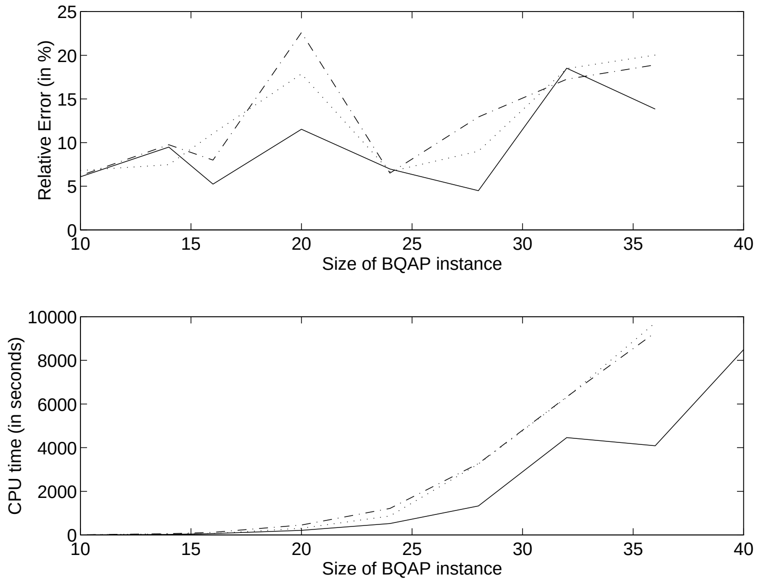


Figure 6.1: *Relative error and CPU time dependence on the problem size. The solid, dotted and dash-dotted curves correspond to Heider's, first and best improvement method, respectively.*

25. For BiQAP instances of size larger than 20, it seems however that only 40 iterations cannot lead to suboptimal solutions of good quality. For BiQAP instances of size $n \leq 20$ the quality of the suboptimal solutions yielded by Tab is much better than the quality of the solutions yielded by Heid, but the corresponding CPU costs are about 5 times higher. We applied Tab twice to all problems P_n involving two different initial permutations, respectively. $3n$ iterations were performed in each run. The result was surprising: in each run Tab produced an optimal solution. Other tests showed that, if the iterations number is larger than $3n$, the quality of the solutions yielded by Tab only slightly depends on the initial permutation.

Next, let us have a look at the performance of LtmTab. In order to not exceed acceptable CPU time limits, LtmTab should run with at most 3 phases, as shown by computational experiments. Trying to preserve reasonable CPU times, we have tested LtmTab with different settings of the control parameters. It results that 20 iterations per phase and tabu length equal to 14 are good choices which yield an acceptable trade off between solution quality and computation time. Figure 6.2 represents the dependence of Tabs and LtmTabs relative error and CPU time on the problem size, respectively. The parameter setting for LtmTab is as described above, whereas 40 iterations per test running of Tab are performed and the length of the involved tabu list equals 25. Figure 6.2 shows the priority of Tab versus LtmTab.

Table 6.2: Performance of TAB

Size	Rel. Error (in %)	CPU (in sec.)	Opt. sol. (in %)
10	0.00	69.50	100.0
14	0.81	348.94	88.9
16	4.71	487.20	82.4
20	11.15	1360.45	42.3
24	3.81	3027.98	21.5
28	7.86	6087.49	13.7
32	17.32	11482.36	4.8

This behavior is due to the quite small number of iterations per phase involved in LTMTAB runnings, as conditioned by CPU time limits. It is however remarkable that in two tests with P_{16} and P_{20}, after two phase runnings with 30 iterations per phase, the solutions produced by LTMTAB were by far better than those produced by TAB. Namely, the corresponding relative error was about 2/3 of the relative error produced by TAB.

Let us now consider the three versions of simulated annealing: ANNEAL, SIMANN2 and SIMANN3. Computational experiments confirm the high sensitivity of ANNEAL with respect to the initial temperature t_0, the number of cooling steps, the equilibrium parameter ϵ and the annealing scheme. On the other hand, the performance of ANNEAL almost does not depend on the parameters N_1 and N_2 determining the maximum number of the attempted transpositions at a fixed temperature state, in the case that their values vary within an appropriate interval. Figures 6.3 and 6.4 provide a graphical illustration of such dependences.

We observed that for a given instance of the problem, there exists no range of initial temperature values which guarantee a relative error within given limits. A basical reason is that ANNEAL's performance strongly depends on the initial permutation. However, for each problem instance there is a corresponding range of initial temperature values which most probably lead to low relative errors. For P_{10} this area is the interval $(2500, 4500)$, as shown in Figure 6.3. This result is also confirmed by other numerical experiments. In most of our test runnings the initial temperature value is derived as follows. Evaluate $\Delta Z = |Z(A,B,\pi) - Z(A,B,\pi')|$ for 100 randomly generated pairs of neighboring permutations (π,π'). The arithmetical mean Δ_m of the corresponding 100 values of Δ is derived and the solution of the following equation is taken as initial temperature value.

$$\exp\left(\frac{-\Delta_m}{t_0}\right) = 0.7 \qquad (6.6)$$

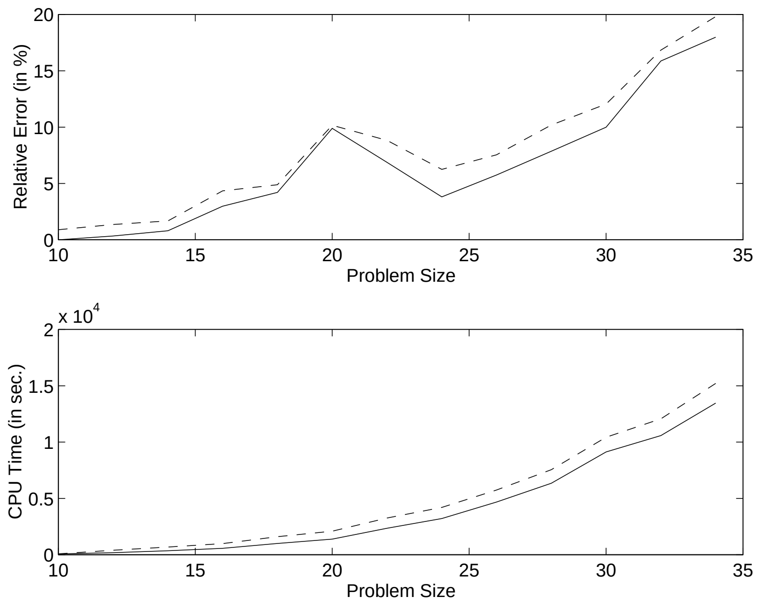


Figure 6.2: *Relative error and CPU time dependence on the problem size. The solid and dashed curves correspond to* Tab *and* LtmTab*, respectively.*

Let us now describe the interesting dependence of Anneal on the amount of cooling steps. For example, Anneal was applied to P16 with initial temperature 7000, epoch length 7 and equilibrium parameter 2000. When the number of cooling steps increased from 10 to 21, the relative error and CPU time decreased and increased respectively. It was surprising that the relative error and the CPU time were almost not influenced by values of the cooling steps number falling out of this range. The reason is that if very few cooling steps are applied the whole search performs at a relatively high temperature and therefore almost all attempted moves are accepted. On the other side, if too many cooling steps are applied there is no further decrease of the relative error. The system reaches very "low" temperatures and only improving moves are accepted. Hence, in this case, Anneal very much approaches a deterministic improvement procedure. After a certain amount of cooling steps, the algorithm stops without dropping through all temperature values, since no further moves are accepted. Consequently, the running times remain almost unchanged when further increasing the number of cooling steps.

In our experiments with Anneal, the temperature schedule is generated by the formula $t_i = 0.7^i \cdot t_0$. For an initial temperature t_0 derived as described above the amount of cooling steps r is defined by $r = \lceil \log_{0.7}\left(\frac{t}{t_0}\right)\rceil$, where t is the solution of the following equation:

$$\exp\left(\frac{-\Delta_m}{t}\right) = 0.2 \tag{6.7}$$

For setting the value of the equilibrium parameter ϵ, 100 pairs of permutations (π, π') are generated and the respective $|\Delta Z| = |Z(A, B, \pi) - Z(A, B, \pi')|$ is computed. Then, the empirical distribution function ΔF of $|\Delta Z|$ is derived. ϵ is defined by solving the equation $\Delta F(\epsilon) = 0.05$. Table 6.3 represents the performance of ANNEAL.

For the second variant of simulated annealing SIMANN2 the temperature schedule is generated by $t_{i+1} = t_i/(1 + \beta \cdot t_i)$. This schedule seems to be more effective than $t_i = q^i \cdot t_0$, $q \in (0, 1)$. This schedule forces the temperature to decrease very fast during the first cooling steps and then, the lower the current temperature is the slower is the cooling of the system. Therefore, it is reasonable to set the initial temperature at a relatively high value in order to accept a large amount of moves at the first cooling steps. Afterwards, most of the search is performed at relatively low temperatures in order to get to good local minima. The initial temperature is defined similarly as for ANNEAL. For Δ_m derived as described above, the initial temperature value is then the solution t_0 of the equation

$$\exp\left(\frac{-\Delta_m}{t_0}\right) = 0.8 \tag{6.8}$$

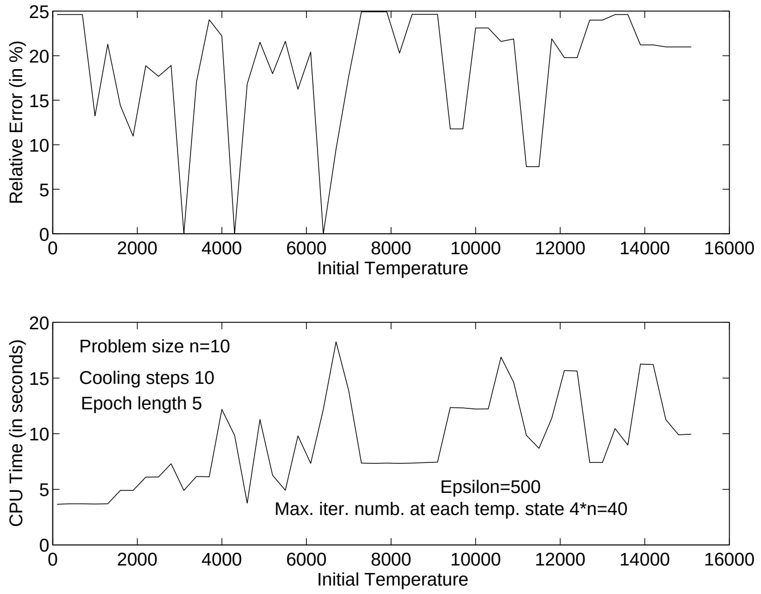


Figure 6.3: *Dependence of* ANNEAL*'s relative error and CPU time on initial temperature* t_0.

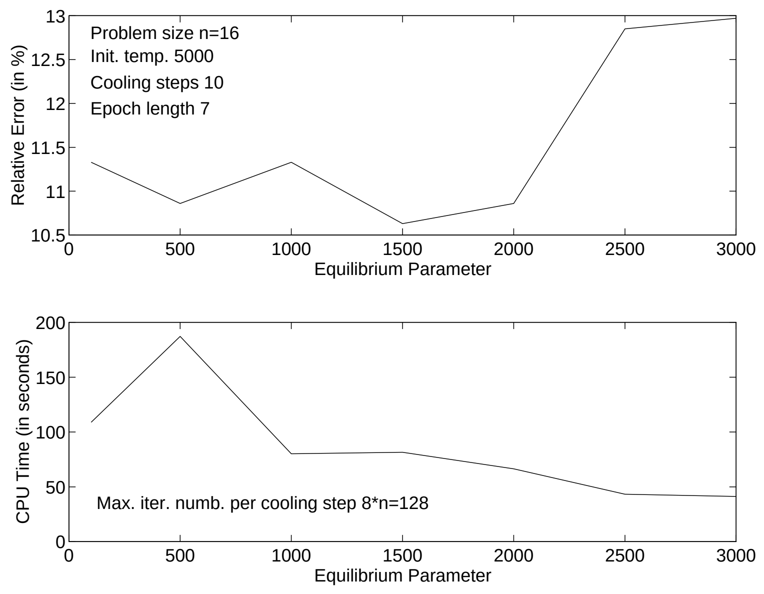


Figure 6.4: *Dependence of* Anneal*'s relative error and CPU time on equilibrium parameter ϵ.*

A good choice for the number of SimAnn2s cooling steps is 10. Further cooling steps yield a quite small improvement of the relative error with high costs of CPU time. Another essential choice is that of the β value. In our tests β is chosen so that the 6 last cooling steps can be performed at temperature values t satisfying the inequality

$$0.01 \leq \exp\left(\frac{-\Delta_m}{t}\right) \leq 0.1 \tag{6.9}$$

The choice of the coefficients at the righthand side of equality (6.8) and at both sides of inequality (6.9) is only empirically and intuitively motivated. Table 6.4 represents the performance of SimAnn2 with at most $n(n-1)/2$ iterations per temperature value, where n is the size of the problem instance.

For the third version of simulated annealing SimAnn3 the temperature schedule is generated by $t_i = q^i t_0$ with $q = 0.8$. The number of iterations at each temperature value during the cooling process and at the optimal temperature should not exceed $n(n-1)/2$, in order to maintain admissible CPU times. Figure 6.5 shows that SimAnn3's performance strongly depends on the initial temperature. This figure summarizes the average results of several runs of SimAnn3 on problem P_{16} with different initial permutations. For all these runs the values of the initial temperature ranged between 1000 and 9500. SimAnn3 found almost always the optimal solution in the case that the percentage of accepted moves at the initial temperature was between

Table 6.3: Control parameter values and performance of ANNEAL

Size	Init. Temp.	ϵ	Cool. Steps	Max. it./ Cool. St.	Rel. Error (in %)	CPU (in sec.)	Opt. sol. (in %)
10	3500	2500	15	90	1.87	11.83	76.7
14	1000	500	11	72	3.64	35.17	22.4
16	4500	3500	13	64	7.62	80.96	4.7
20	6000	7000	7	140	14.04	252.72	5.3
24	6000	6000	8	144	7.90	291.90	3.2
28	12000	15000	10	188	13.62	849.99	2.8
32	15000	16000	10	256	15.36	2061.65	2.1

Table 6.4: Control parameter values and performance of SIMANN2

Size	Init. Temp.	β	Rel. Error (in %)	CPU (in sec.)	Opt. sol. (in %)
10	1000	0.0010	0.61	6.50	98.7
14	900	0.0010	6.00	32.50	20.5
16	1000	0.0005	6.75	69.25	23.8
20	3000	0.0001	11.00	149.75	49.4
24	3000	0.0001	8.25	70.50	5.7
28	5000	0.0001	13.25	686.75	23.8
32	7000	0.0005	19.00	818.50	4.3

43% and 47%. Similar results for instances of different sizes suggest the following empirical "rule" for setting the value of the initial temperature: The percentage of the accepted permutations at the initial temperature should be between 43% and 47%. For such initial temperature values about 10 cooling steps are sufficient for a suboptimal solution of good quality. This empirical rule has the following intuitive basis:

1. After the search diversification realized at the first phase, only "good" solutions should be accepted at the second phase. This suggests an optimal temperature choice which yields a relatively small acceptance ratio.

2. If the values of q, the amount of cooling steps and the initial temperature are as described above, the acceptance ratio at the respective optimal temperature

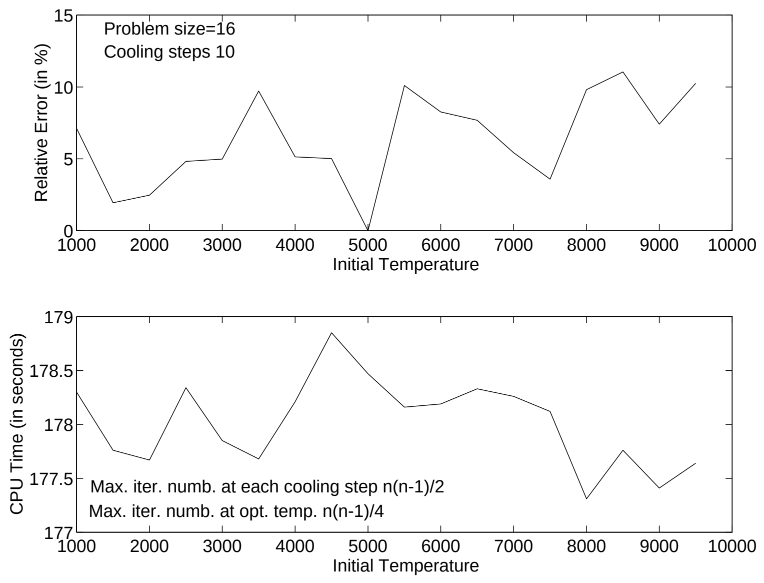


Figure 6.5: *Dependence of* SIMANN3*'s relative error and CPU time on the initial temperature value.*

ranges between 11% and 17% .

For initial temperature values which do not match the rule described above the increase of the cooling steps number usually leads to a decrease of the relative error. Finally, the initial temperature and the indices of the SIMANN3's performance (relative error and CPU time) are represented in Table 6.5. Results on a comparison ANNEAL, SIMANN2 and SIMANN3 are summarized in Figure 6.6

TABSIM (tabu search combinated with simulated annealing) involves five control parameters: initial temperature, cooling steps number, iteration number per cooling step, tabu length and iteration number in the second phase (tabu phase). Similar arguments as in the case of SIMANN3 suggest the same choice for the values of the three first control parameters, whereas the values of the two last parameters cannot be adopted from TAB; this would lead to infeasible CPU times. Computational experiments showed that a tabu list of length 15 and 20 iterations in the second phase leads to a relatively good trade off between computation time and solution quality. An increase of the iteration number yields slightly better solutions at pretty high costs of CPU time. Table 6.6 summarizes some results on the TABSIM's performance. The results of this table confirm the bad performance of TABSIM in comparison with SIMANN3. Such a behavior is due to the fact that the amount of 20 iterations, conditioned by growing computation time requirements, are insufficient in the second phase of the method. We applied TABSIM to P_{28} with 35 iterations at the second phase

Table 6.5: Control parameter values and performance of SIMANN3

Size	Init. Temp.	Rel. Error (in %)	CPU (in sec.)	Opt. sol. (in %)
10	2000	0.09	14.00	72.1
14	3000	0.43	86.26	98.4
16	5000	0.67	177.16	95.6
20	7000	1.05	579.09	93.2
24	10000	5.37	1489.50	23.5
28	16000	8.65	3289.23	8.9
32	20000	4.61	6978.59	78.9

Table 6.6: Performance of TABSIM

Size	Rel. Error (in %)	CPU (in sec.)	Opt. sol. (in %)
10	0.00	43.65	100.0
14	2.82	232.95	79.5
16	4.81	442.81	59.7
20	5.54	1290.31	96.5
24	6.35	3398.93	18.3
28	8.76	6887.23	11.7
32	11.87	11740.32	6.3

and got an average relative error of 4.11% instead of 8.65% obtained by SIMANN3. However, this improved relative error costs twice as much CPU time. It turns out that the use of TABSIM might be quite effective for problems to which SIMANN3 is once applied. In this case, a higher iteration number at the second phase of the algorithm would be feasible, since the optimal temperature value is already computed by SIMANN3. Clearly, the best solution produced by SIMANN3 can be used as initial solution for the second phase of TABSIM.

Concluding, Figure 6.7 represents the graphical comparison of HEID, TAB and SIMANN3, judged to be the best among improvement methods, tabu search methods and simulated annealing methods for BiQAPs, respectively.

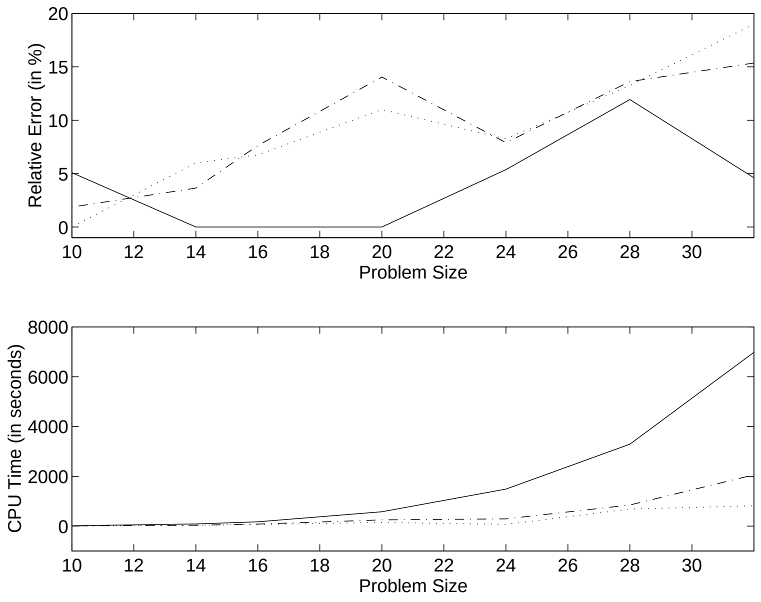


Figure 6.6: *Relative error and CPU time dependence on the problem size. The solid, dotted and dash-dotted curves correspond to* SimAnn3*,* SimAnn2 *and* Anneal*, respectively.*

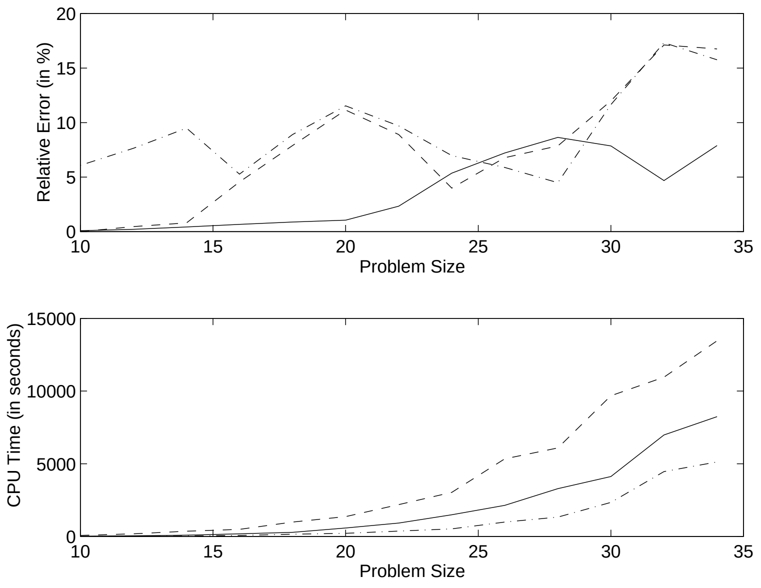


Figure 6.7: *Relative error and CPU time dependence on the problem size. The solid, dashed and dash-dotted curves correspond to* SimAnn3*,*Tab *and* Heid*, respectively.*

6.4 Conclusions and open question on BiQAPs

In the last two chapters we investigated a generalization of the quadratic assignment problem, namely the BiQAP. This is a new problem of practical relevance which arises in VLSI design. There exists a strong relationship between the QAP and the BiQAP, in the sense that each QAP can equivalently be formulated as a BiQAP. This fact obviously implies the BiQAP to be at least as hard as the QAP from both the theoretical and the practical point of view. Moreover, the input size of a BiQAP of size n is $O(n^4)$, whereas the input size of a QAP of size n is $O(n^2)$. Clearly, this can lead to difficulties when practically dealing with the problem.

From a theoretical point of view, our contribution concerns two aspects. First, generalizing the results already known for QAPs, a 0-1 and a mixed linear programming formulation for BiQAPs are derived, respectively. It is not surprising that the size of these linearizations explodes as the size of the BiQAP increases. Secondly, the asymptotical behavior of BiQAPs is analyzed. It turns out that, under natural probabilistic constraints, the problem becomes "easy" as its size tends to infinity. That is, its best and worst objective function values get closer as the size of the problem increases. From a practical point of view this means that for problems of very large size each feasible solution is close enough to an optimal one. Deriving a lower bound for the size of instances which show such an asymptotic behavior remains an interesting and apparently difficult open problem.

From a practical point of view, our results approach three main directions. First, lower bounds for BiQAPs are derived by generalizing the Gilmore-Lawler bounds for QAPs. As it is not easy to identify a nice algebraic or combinatorial structure for the BiQAP, Gilmore-Lawler type bounds seems to be the only straightforward bounding procedure for this problem. Secondly, a generator of BiQAP instances with known optimal solution is proposed. It produces instances which are combinatorially related to QAPs with known optimal solution, generated as proposed by Li and Pardalos [117]. These BiQAP instances are used for testing the proposed bounding procedure and, further on, the performance of several heuristics for BiQAPs. It turns out that the quality of the lower bounds is not satisfactory, especially when the size of the problem increases. At this point an interesting open question arises, namely: How difficult are the generated test instances? The relevance of this open question is twofold. Theoretical - concerning the computational complexity of the test examples, and practical - concerning their so called *average complexity*.

Our third contribution on BiQAPs is related to heuristic approaches. Several version of deterministic improvement methods, simulated annealing and tabu search are proposed and tested, providing also a detailed analysis of their performance and their sensitivity with respect to different control parameters. This analysis focuses on the comparison of the heuristics performance when applied to test problems with known optimal solution. It turns out that the third version of simulated annealing, SIMANN3, yields the lowest values of the relative error for almost all considered

BiQAP instances. The CPU time requirements of SIMANN3 are slightly higher than those of HEID, which turned out to be the best among the proposed deterministic improvement methods. However, in most of the cases, HEID's relative error is at least twice as much as the relative error yielded by SIMANN3. Both SIMANN3 and HEID outperform the tabu search version TAB: for the same CPU time costs, the relative error yielded by TAB is in average $1.5 - 2$ times the error of the two first methods. On this basis, we conclude that among the heuristics tested in this chapter the third version of simulated annealing SIMANN3 provides the best results. However, let us emphasize the need for a better trade off between solution quality and computation time. In this frame, we are experimenting with a more robust version of tabu search, namely the reactive tabu search [10], and the preliminary results are very promising. Another probably interesting research direction is the investigation of BiQAPs whose coefficient arrays have some special structure. For example, BiQAP instances which occur in practical applications as described in Section 5.1 have very sparse coefficient arrays. Is it possible to exploit the structure of the coefficient arrays for deriving better heuristics for this kind of BiQAPs? Recall that none of the methods proposed in this chapter makes use of specific combinatorial properties of the problem, apart from the inherent neighborhood structure of its feasible solutions set.

Chapter 7

The Communication Assignment Problem (CAP)

In this chapter we deal with the so called *communication assignment problem (CAP)* introduced in Subsection 1.3.6. CAP arises when locating stations in a local area computer network (LAN) with the aim that the information "load" going through the busiest station is minimized. In the first section we describe in detail this practical problem and show how to mathematically formulate it as a CAP. In this frame, two models of the CAP are distinguished: *the single path* model and the *fractional* model.

When investigating the computational complexity of the CAP, it turns out that even for simple underlying graphs (for example, paths and cycles) the CAP is NP-hard. However, polynomially solvable cases of the problem can be distinguished. Namely, in the cases that the underlying graph is a star of branch length 2 or a doublestar, the CAP becomes polynomially solvable in the single path model and in the fractional model, respectively. Further, it is shown that an optimal solution of the general CAP can be found by solving several (exponentially many, in general) linear programs. Among the optimal solutions of these linear programs it is chosen one which yields the smallest value of the corresponding objective function. This is also the optimal solution of the CAP. This approach can sometimes be helpful to distinguish polynomially solvable cases of the CAP, although the number of the linear programs to be solved is in general exponential. Complexity results on the CAP are summarized in the third section.

Given the NP-hardness of the problem even for simple graph classes, we propose a branch and bound algorithm for the case when the underlying graph is a tree. However, the lower bound computation is very expensive, and thus prohibitive, when trying to solve problems of size larger then 11. This branch and bound approach together with some computational results are described in the second section.

Finally, we investigate the asymptotic behavior of the CAP on trees. It shows the same asymptotical behavior as the QAP and the BiQAP. Namely, under natural probabilistic assumptions, the ratio between the "worst" and the "best" objective function value tends to one as the size of the problem tends to infinity. A correct mathematical formulation of this result and its proof are provided in Section 7.4.

Concluding, a summary of the achieved results and open questions on the CAP follows in Section 7.5.

7.1 A practical application of the CAP

A system $\mathcal{C}$ of n communication centers $C_1, \ldots, C_n$ has to be embedded into an undirected graph $G = (V, E)$, where $V = \{1, 2, \ldots, n\}$. Each communication center C_i transmits messages to every other center C_j at a rate of t_{ij} messages per time unit ($t_{ii} \equiv 0$, for $i \geq 1$). Recall that $T = (t_{ij})$ is called *communication matrix*. Moreover, all communication matrices in this chapter have nonnegative entries. In the case that there is no direct connection between centers C_i and C_j, the messages are routed from one center to the other via several intermediate centers. Recall that an embedding of the centers into the vertices of the graph V is mathematically represented by a permutation π of $\{1, 2, \ldots, n\}$, $\pi \in \mathcal{S}_n$, where $\pi(i)$ is the index of the center embedded at vertex $v_i \in V$. From now on we identify vertices v_i and centers C_i with their indices, respectively. Recall moreover that $|conn(i,j)|$ is the number of paths in G connecting vertices i and j and that a fixed order among these paths is considered. Namely, we have the following sequence of paths:

$$conn(i,j) = \left\langle P_1^{(i,j)}, \ldots, P_{|conn(i,j)|}^{(i,j)} \right\rangle$$

Further, a routing pattern shows how the messages between every pair of centers are routed through the graph. For a fixed embedding π, a feasible routing pattern ρ maps each pair (i,j) of vertices to a vector of size $|conn(i,j)|$, $\left(y_1^{(i,j)}, \ldots, y_{|conn(i,j)|}^{(i,j)}\right)^t$, such that

$$\sum_{k=1}^{|conn(i,j)|} y_k^{(i,j)} = t_{\pi(i)\pi(j)}$$

For a pair (π, ρ), where π is an embedding and ρ is a routing pattern, $N_{\pi,\rho}(k)$ denotes the overall amount of traffic going through the center $C_{\pi(k)}$ as intermediate center. The goal is to find an embedding π and a routing pattern ρ that minimizes the maximum intermediate traffic over all centers. Thus, CAP(T,G) can be formulated as:

$$\min_{\pi,\rho} \max\{N_{\pi,\rho}\colon 1 \leq i \leq n\} \tag{7.1}$$

If (π, ρ) is a pair which minimizes the objective function, i.e., is an optimal solution to (7.1), π is termed *optimal embedding* and ρ is termed *optimal routing pattern*. The corresponding maximum noise is called optimal value. Essentially, we distinguish two models of routing patterns. In the *single path model*, for every pair of communication centers C_i and C_j, a single route in the network is selected and all messages t_{ij} are sent along this fixed route. In the *fractional model*, the amount t_{ij} is split into a number of (positive) parts and every part is sent along its own path. Clearly, in the case that the underlying graph is a tree both models coincide.

A typical application of the problem arises when stations (terminals, computers) are to be located in a local-area computer network (LAN). In order to prevent interference and hence garbled messages, only one message at a time has to be sent through a given segment of LAN. To gain access to the segment a given station:

1. waits until the segment is detected as being free
2. may wait a further random time t.
3. begins to transmit its messages.

Consider the LAN represented by Figure 7.1. The stations 1 and 2 which have detected the segment a, b as being free, start to transmit messages after having waited random times t_1 and t_2, respectively. Since t_1 and t_2 are simply random values and since the messages do not pass through the segment instantaneously, conflicts may occur. In this case both stations back off the messages and process through the steps $1, 2, 3$ again. Conflicts result in wasted time and delays in messages being sent. As the offered traffic increases a point is reached when the conflicts become so frequent that the throughput begins to decrease. This problem is described in details by Stallings [170].

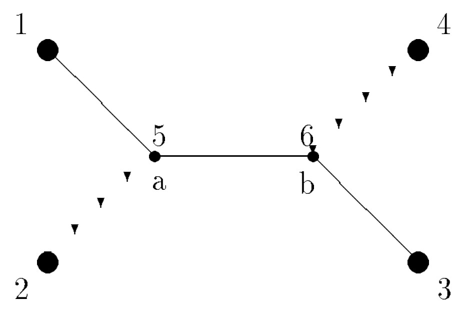


Figure 7.1: A segment of LAN with bridges

In order to restrict the offered traffic through the same segment of LAN , it is reasonable to locate *bridges* at its endpoints. For example, the bridges 5 and 6 are located at nodes a and b, respectively. These bridges will work as intermediate centers. In this case, if 2 has to send a message to 4, it does not need to wait until the segment a, b becomes free. 2 sends its message to 5 and 5 cares to forward the message to 6 and so on, until the message reaches its destination. If all stations work as bridges,

the result is that each pair of stations (or bridges) communicates through its own segment. It is reasonable to require an embedding of stations and additional bridges into the nodes of LAN such that the intermediate traffic going through the busiest station (or bridge) is minimized. Boffey and Karkazis [16] proposed and discussed a continuous version of the problem.
A similar problem, the so called *elevator problem* is described by Karkazis [99]. In the same paper Karkazis describes a branch and bound algorithm for solving problem (7.1) in the case that G is a path.

If we consider the problem of minimizing the *sum* of the intermediate traffic going through all centers in the case that G is a tree, we get a quadratic assignment problem (QAP). One of its coefficient matrices is the matrix of the lengths of paths joining each pair of vertices in the given tree, whereas the other one is the communication matrix $T = (t_{ij})$. The optimal solution of this QAP does not yield, however, an adequately low maximal intermediate traffic when applied to problem (7.1) (see Stallings [170]).

7.2 A branch and bound approach for the CAP on trees

In this section we consider the CAP on trees and derive a branch and bound algorithm for this problem. It turns out that it is convenient to work with a rooted tree. This facilitates the computation of lower bounds and the search in a branch and bound approach for the CAP. Hence, we first show how to appropriately define a root in a given tree.

7.2.1 Definitions and observations

Let $G = (V, E)$ be a tree, where V is the set of its vertices, $|V| = n$, and E is the set of its edges. An edge connecting two vertices $u, v \in V$ is denoted by (u, v). Assume that for each vertex $v \in V$ we are given the set of its adjacent vertices $A(v)$. For any $v \in V$, $d(v)$ denotes the degree of v, i.e. $d(v) = |A(v)|$. Moreover, we assign a level $l(i)$ to each of the vertices v_i, $1 \le i \le n$, of V. This is done recursively as follows. Assign level 1 to all leaves of G. Then, delete all vertices of level 1 and their incident edges. The remaining graph is obviously again a tree. Assign level 2 to all leaves in this new tree and again delete all its leaves with their incident edges. Iteratively apply the above described procedure for the current tree, where the value of the level assigned to its leaves is increased by one in each step. Then, the vertices which got their levels at last are deleted together with their incident edges and the current tree is updated. The process of assigning levels to the vertices of a tree as described above is called *shelling* of the tree. It is easy to see that in the last iteration of a shelling process either one vertex or two vertices are left. The latter case can always be transformed to the first one by adding a dummy vertex and two dummy edges to the given tree G. Namely, let us assume that two vertices, say x and y, get their levels at the last iteration. Clearly, this vertices are connected by an edge, i.e., $(x, y) \in E$ (otherwise G would not be a tree). We add a dummy vertex to V, say $V := V \cup \{z\}$. Moreover, we delete edge (x, y) and add two dummy edges (x, z) and (y, z), $E := E \setminus \{(x, y)\} \cup \{(x, z), (y, z)\}$. Let us denote the transformed tree by G'. Clearly, the new graph G' is a tree and its shelling terminates with a single vertex, namely with z. As z is simply a dummy vertex, the center corresponding should not communicate with the other centers. Let us denote by T' a $(n+1) \times (n+1)$ communication matrix obtained from the $n \times n$ matrix T by adding to it an $(n+1)$-th row and an $(n+1)$-th column each of them consisting only of zeros. It is clear that CAP(T,G) and CAP(T$'$,G$'$) are equivalent, in the sense that the solution for one of the problems can be transformed in polynomial time to the solution of the other and vice-versa. Under these conditions, it is no loss of generality to consider only instances CAP(T,G) such that the shelling of G ends up with only one vertex. From now on we assume that all the trees considered in this section have this property. Then, the vertex which gets its level at last is defined to be the *root* of the tree. The level of the root is called *level of the tree* and is denoted by λ. Moreover, we define the *level sets* L_i, $1 \le i \le \lambda$, where set L_i consists of the vertices of V whose level is equal to i. The following observations, whose proofs are omitted because of their simplicity, summarize properties of level sets.

Observation 7.1 *The level sets L_i have the following properties.*
1. $u, v \in L_i$, $u \neq v$, $i = 1, 2, \ldots, \lambda - 1 \Longrightarrow (u, v) \notin E$. ■

Observation 7.2 *Let a tree G with level λ be given. If $v \in V$ with $l(v) = i$, the following holds:*

$$|A(v) \cap \bigcup_{j=i}^{\lambda} L_j| = 1, \textit{ for } i < \lambda, \quad \textit{and} \quad A(v) \cap L_{i-1} \neq \emptyset, \textit{ for } i > 1. \tag{7.2}$$

■

Once we have distinguished a root in G, *fathers*, *sons* and *successors* can be defined as usually.

Definition 7.1 *Let $G = (V, E)$ be a tree with level λ and $|L_\lambda| = 1$. Consider a vertex $v \in V$ with $l(v) = i < \lambda$. The vertex $u \in V$ such that $\{u\} = A(v) \cap (\bigcup_{j=i}^{\lambda} L_j)$ is the* father *of v and will be denoted by $f(v)$. The elements of $A(v) \setminus \{f(v)\}$ are the* sons *of v and the set of sons of a given vertex v is denoted by $S(v)$. The set $R(v)$ of the* successors *of v is given then as follows:*

$$R(v) = \left\{ u \colon u \in \bigcup_{i=1}^{l(v)-1} L_i \textit{ and } \forall\, k \in P(u, v), k \in \bigcup_{i=1}^{l(v)-1} L_i \right\} \bigcup \{v\} \ ,$$

where $P(u,v)$ is the path joining the vertices u and v in the tree G, and $k \in P(u,v)$ means that k is an inner vertex of $P(u,v)$.
The root v_0 of the tree ($L_\lambda = \{v_0\}$) has no father and all its adjacent vertices are its sons, $S(v_0) = A(v_0)$. Clearly, $R(v_0) = V$.

Let us illustrate these notations in the undirected tree represented by Figure 7.2. Its vertices are labeled by numbers from 1 to 13. The arrows show the directions father-son with respect to our definitions. The level sets are as follows

$$L_1 = \{5,6,8,9,10,11,12\} \quad L_2 = \{2,3,7\} \quad L_3 = \{4,1\} \quad L_4 = \{13\}$$

The root of the tree is the vertex labeled by 13 and the tree has level 4.

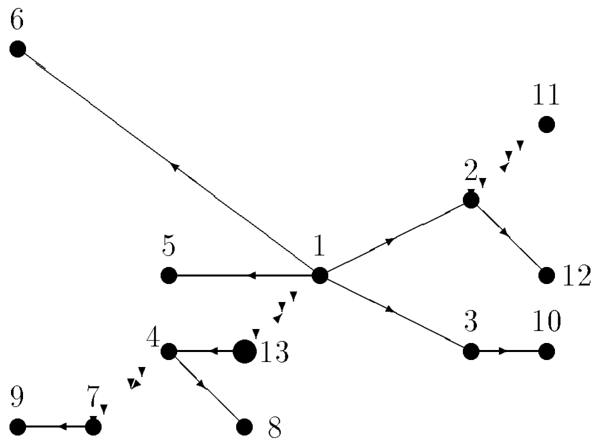


Figure 7.2: Shelling a tree: an illustrative example

Now, let us return to CAP(T,G), where $G = (V,E)$ is a tree and $|V| = n$. Let an embedding $\pi \in \mathcal{S}_n$ of the communication centers into the vertices of the tree be given. The only way to transmit messages from the center $C_{\pi(i)}$ to the center $C_{\pi(j)}$ placed at the vertices i and j, respectively, is the unique path $P(i,j)$ in the tree which joins the vertices i and j. Recall that $k \in P(i,j)$ iff the path $P(i,j)$ contains the vertex k as inner vertex. A message sent from center $C_{\pi(i)}$ to center $C_{\pi(j)}$ will pass through center $C_{\pi(k)}$ iff $k \in P(i,j)$. The noise $N_\pi(k)$ at vertex k, that is the total amount of traffic going through $C_{\pi(k)}$ as an intermediate center with respect to the given embedding π, is then given as

$$N_\pi(k) = \sum_{(i,j):k\in P(i,j)} t_{\pi(i)\pi(j)} \tag{7.3}$$

Through the rest of this section, the objective function of the CAP(T,G) corresponding to an embedding π will be sometimes denoted by $CAP(T,G,\pi)$:

$$CAP(T,G,\pi) = \max\{N_\pi(k): 1 \le i \le n\}$$

When trying to represent the noise by a more appropriate formula the following observation will be useful. Again we omit the proof because of its simplicity:

Observation 7.3 *Given a tree $G = (V,E)$ and three pairwise distinct vertices $i,j,k \in V$. The following statements are equivalent:*

1. *$k \in P(i,j)$.*
2. *Either there exist $l,t \in S(k)$, $l \neq t$, such that $i \in R(l)$ and $j \in R(t)$ or $|\{i,j\} \cap R(k)| = 1$.* ■

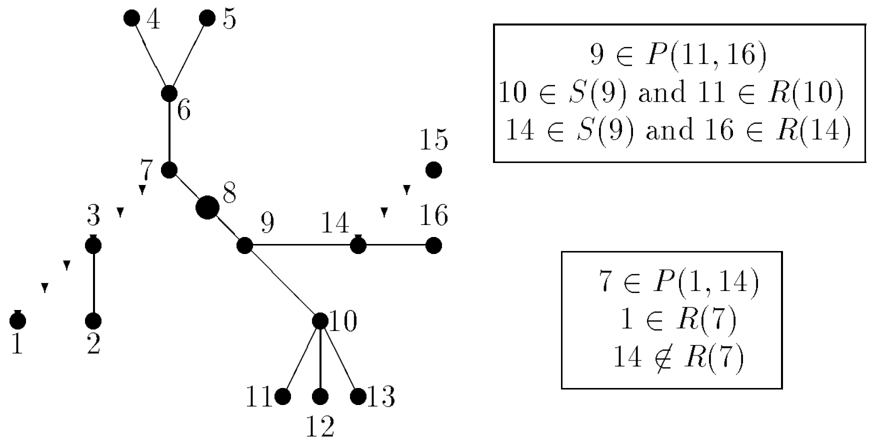


Figure 7.3: An illustration for Observation 7.3

7.2.2 Lower bounds

For a tree G on n vertices and an $n \times n$ communication matrix $T = (t_{ij})$, CAP(T,G) can be formulated as follows:

$$\min_{\pi \in \mathcal{S}_n} CAP(T,G,\pi) \tag{7.4}$$

In order to simplify and shorten the notations, the noise $N_\pi(i)$, $1 \le i \le n$, $\pi \in \mathcal{S}_n$, is divided into two parts $N_\pi^{(1)}(i)$ and $N_\pi^{(2)}(i)$ which comply with its construction:

$$N_\pi^{(1)}(i) := \sum_{\substack{j,k \in S(i) \\ j \neq k}} \sum_{p \in R(j)} \sum_{t \in R(k)} t_{\pi(p)\pi(t)} \tag{7.5}$$

$$N_\pi^{(2)}(i) = \sum_{j\in R(i)\setminus\{i\}} \sum_{k\notin R(i)} \Big(t_{\pi(j)\pi(k)} + t_{\pi(k)\pi(j)}\Big) \tag{7.6}$$

$N_\pi^{(1)}(i)$ represents the overall amount of messages exchanged between all pairs of centers $(C_{\pi(p)}, C_{\pi(t)})$ embedded at vertices p and t, where p and t are *successors of distinct sons of* i, respectively. The sum $N_\pi^{(2)}(i)$ represents the overall amount of messages exchanged between pairs of centers $C_{\pi(j)}$ and $C_{\pi(k)}$ embedded at the vertices j and k, where j is a successor of i, $j \neq i$, and k it is not a successor of i. According to Observation 7.3 and to equality (7.3) we get

$$N_\pi(i) = N_\pi^1(i) + N_\pi^2(i) \tag{7.7}$$

For a given $A \subset \{1,2,\dots,n\}$ a *partial embedding* is a one to one mapping $\pi\colon A \to \{1,2,\dots,n\}$. Given a partial embedding π_0 and a subset A as above, we use the following notation

$$\mathcal{S}_n^A = \{\pi \in \mathcal{S}_n \colon \pi(i) = \pi_0(i),\ \forall i \in A\} \tag{7.8}$$

Obviously, $\mathcal{S}_n^A$ is specific to the partial embedding π_0. But, as it is always easy to realize which is the partial embedding π_0 related to the concrete $\mathcal{S}_n^A$, our notation does not reflect this relationship. Moreover, for a given set $D \subset \{1,2,\dots,n\}$ we denote by D_A the intersection $D \cap A$.
In the following a lower bound for the so called *partial problem*

$$\min_{\pi\in\mathcal{S}_n^A} CAP(T,G,\pi). \tag{7.9}$$

is computed. For a given $\pi \in \mathcal{S}_n^A$ and $i = 1,2,\dots,n$, let $B_\pi^{(1)}(i)$, $B_\pi^{(2)}(i)$ and $B_\pi(i)$ be defined as follows:

$$B_\pi^{(1)}(i) = \sum_{\substack{j,k\in S(i)\\ j\neq k}} \sum_{p\in R_A(j)} \sum_{t\in R_A(k)} t_{\pi(p)\pi(t)} \tag{7.10}$$

$$B_\pi^{(2)}(i) = \sum_{j\in R_A(i)\setminus\{i\}} \sum_{k\in A\setminus R(i)} \Big(t_{\pi(j)\pi(k)} + t_{\pi(k)\pi(j)}\Big) \tag{7.11}$$

$$B_\pi(i) = B_\pi^{(1)}(i) + B_\pi^{(2)}(i) \tag{7.12}$$

$B_\pi^{(1)}(i)$ is the overall amount of messages exchanged between centers $C_{\pi(p)}$ and $C_{\pi(t)}$ embedded at the vertices p and t, where p,t are successors of different sons of i and, additionally, $p,t \in A$. The sum $B_\pi^{(2)}(i)$ is the overall amount of messages exchanged between centers $C_{\pi(j)}$ and $C_{\pi(k)}$ embedded at the vertices j and k, where j is a successor of i, $j \neq i$, k is not a successor of i and, additionally, $j,k \in A$. Notice that $B_\pi^{(1)}(i)$, $B_\pi^{(2)}(i)$ and $B_\pi(i)$, $i = 1,2,\dots,n$, do not depend on the permutation $\pi \in \mathcal{S}_n^A$, i.e., for all $\pi \in \mathcal{S}_n^A$, $B_\pi^{(j)}(i) = B_{\pi_0}^{(j)}(i)$, $j = 1,2$, $i = 1,2,\dots,n$, where π_0 is the partial permutation related to $\mathcal{S}_n^A$. Moreover, the following inequalities obviously hold:

$$N_\pi^{(1)}(i) \geq B_\pi^{(1)}(i) \quad \text{and} \quad N_\pi^{(2)}(i) \geq B_\pi^{(2)}(i)$$

(Observe that $B_\pi^{(1)}(i)$, $B_\pi^{(2)}(i)$ are derived by summing up a part of the terms summed up at $N_\pi^{(1)}(i)$, $N_\pi^{(2)}(i)$, respectively, and remember that all these terms are positive.) Thus, for $1 \le i \le n$, $B_{\pi_0}(i)$ is a lower bound for $N_\pi(i)$, for all $\pi \in \mathcal{S}_n^A$. The following observation enables us to improve the quality of this lower bound.

Observation 7.4 *Let $A \subseteq \{1, 2, \ldots, n\}$ be given. If $i \in A$ and $R(i) \subseteq A$, then $N_\pi(i) = N_{\pi_0}(i)$, for all $\pi \in \mathcal{S}_n^A$.*

Proof. If $i \in A$ and $R(i) \subseteq A$, then $R(k) \subseteq R(i) \subseteq A$, $\forall k \in S(i)$. Considering that $\pi(j) = \pi_0(j)$, $\forall j \in A$, the equality $N_\pi^{(1)}(i) = N_{\pi_0}^{(1)}(i)$ is obviously true. Moreover, $\forall \pi \in \mathcal{S}_n^A$, the following equalities hold:

$$\begin{aligned} N_\pi^{(2)}(i) &= \sum_{j \in R(i) \setminus \{i\}} \sum_{k \notin R(i)} \left(t_{\pi(j)\pi(k)} + t_{\pi(k)\pi(j)} \right) \\ &= \sum_{j \in R(i) \setminus \{i\}} \sum_{k \notin \pi_0(R(i))} \left(t_{\pi_0(j)k} + t_{k\pi_0(j)} \right) = N_{\pi_0}^{(2)}(i) \qquad \blacksquare \end{aligned}$$

Observation 2.4 shows that for all $i \in A$ such that $R(i) \subseteq A$, the noise $N_\pi(i)$ can be computed exactly, namely $N_\pi(i) = N_{\pi_0}(i)$, $\forall \pi \in \mathcal{S}_n^A$. The following lemma gives a lower bound for $N_\pi(i)$, $\pi \in \mathcal{S}_n^A$, in the case that $i \notin A$, but $(R(i) \setminus \{i\}) \subseteq A$. In order to shorten the notation let us denote $R^-(i) = R(i) \setminus \{i\}$.

Lemma 7.5 *Let $A \subseteq \{1, 2, \ldots, n\}$ be given. If $i \notin A$ and $R^-(i) \subseteq A$, then*

$$\begin{aligned} N_\pi(i) \ge B_{\pi_0}^{(1)}(i) \quad &+ \sum_{k \notin \pi_0(R^-(i))} \sum_{j \in R^-(i)} \left(t_{\pi_0(j)k} + t_{k\pi_0(j)} \right) \\ &- \max_{k \notin \pi_0(A)} \sum_{j \in R^-(i)} \left(t_{\pi_0(j)k} + t_{k\pi_0(j)} \right) \end{aligned} \tag{7.13}$$

Proof.
If $i \notin A$ and $R^-(i) \subseteq A$, the inclusion $R(j) \subseteq A$ holds for all $j \in S(i)$. Therefore

$$N_\pi^{(1)}(i) = B_{\pi_0}^{(1)}(i), \quad \forall \pi \in \mathcal{S}_n^A \tag{7.14}$$

Moreover, taking into account that $\pi(A) = \pi_0(A)$, $\forall \pi \in \mathcal{S}_n^A$, it is easily seen that the following inequality holds:

$$N_\pi^{(2)}(i) + \max_{k \notin \pi_0(A)} \sum_{j \in R^-(i)} \left(t_{\pi_0(j)k} + t_{k\pi_0(j)} \right) \ge \sum_{j \in R^-(i)} \sum_{k \notin \pi_0(R^-(i))} \left(t_{\pi_0(j)k} + t_{k\pi_0(j)} \right)$$

Summing up side by side the last inequality and equality (7.14) completes the proof. $\blacksquare$

The expression in the righthand side of inequality (7.13) will be denoted by $B_{\pi_0}^*(i)$, $i = 1, 2, \ldots, n$.

We wish to sharpen the bound $B_{\pi_0}(i)$ also for $i \notin A$ such that $R^-(i) \nsubseteq A$. Through the rest of this subsection, let a set $A \subset V$ with the property

$$(i \in A) \Rightarrow R(i) \subseteq A \tag{$**$}$$

and a $\pi_0 \in \mathcal{S}_n$ be given and fixed. Moreover, the following definitions will be useful:

Definition 7.2 *For a given $k \in V$, the number of paths which contain k as an inner vertex is called* index *of k. This index is denoted by $\mathcal{I}(k)$.*

Definition 7.3 *The* active index *of $k \in V$ with respect to A is the number of paths whose endpoints are not both elements of A and which contain k as an inner vertex. The active index is denoted by $\mathcal{I}_a(k)$.*

Definition 7.4 *Consider the entries t_{ij}, $i \neq j$, of the communication matrix T such that either $i \notin \pi_0(A)$ or $j \notin \pi_0(A)$ and sort them increasingly. We denote the resulting vector by $\Omega^A_{\pi_0}$ and we call it* Ω-vector of T with respect to A and π_0.

Definition 7.5 The additive bound *at $k \in V$* with respect to A and π_0 *is the sum of the $\mathcal{I}_a(k)$ first components of the Ω-vector $\Omega^A_{\pi_0}$. We denote it by $B^+_{\pi_0}(k)$.*

It is easily seen that the difference $N_\pi(i) - B_{\pi_0}(i)$ consists of a sum of $\mathcal{I}_a(i)$ components of the Ω-vector $\Omega^A_{\pi_0}$. Hence, the following inequality holds and it allows us to replace $B_{\pi_0}(i)$ by the sharper bound $B'_{\pi_0}(i)$:

$$N_\pi(i) \geq B_{\pi_0}(i) + B^+_{\pi_0}(i) = B'_{\pi_0}(i) \tag{7.15}$$

We can now define a lower bound to be used for our branch and bound algorithm. We introduce the following notations:

$$B^* := \max\left\{B^*_{\pi_0}(i) \colon (i \notin A) \wedge (R^-(i) \subseteq A)\right\}$$

$$B' := \max\left\{B'_{\pi_0}(i) \colon (i \notin A) \wedge (R^-(i) \nsubseteq A)\right\}$$

$$N := \max\left\{N_{\pi_0}(i) \colon i \in A\right\}$$

and

$$B := \max\left\{B^*, B', N\right\}$$

The following inequality represents the main result of this section:

$$\min_{\pi \in \mathcal{S}^A_n} CAP(T, G, \pi) \geq B \tag{7.16}$$

For the given undirected tree we use the data structure described in Subsection 7.2.1, i.e., for each vertex we save its father and the set of its sons. The size of the problem data is $O(n)$. For each vertex $i \in V$ we derive the set $R(i)$ of its

successors in $O(n)$ time. Then, for given A and π_0, we derive $N_{\pi_0}(i)$, or $B'_{\pi_0}(i)$ or $B^*_{\pi_0}(i)$, $1 \le i \le n$, (depending on the position of i with respect to A) in $O(n^2)$ time. The maximum of these values over $1 \le i \le n$ is found in $O(n)$ time. Summarizing, we need $O(n^3)$ time for computing a lower bound for the partial problem (7.9). As shown in Subsection 7.2.4, this lower bound computation can be speeded up, when depth first search (DFS) is involved in the branch and bound algorithm.
An appropriate choice of a suitable $A \subset V$, i.e., which fulfills the condition $(**)$, involves the level sets of the given tree. The level sets, the level of each vertex and the root of the tree can be derived in $O(n^2)$ time, by applying the recursive procedure described in Subsection 7.2.1.

7.2.3 The description of the algorithm

We assign the best embedding found so far and the corresponding value of the objective function to the variables π_{opt} and Z_{opt}, respectively.
Each node α of the branch and bound tree (BBT), is characterized by a set A_α, $A_\alpha \subseteq \{1,2,\dots,n\}$, and by a partial embedding $\pi_\alpha\colon A \to \{1,2,\dots,n\}$. Such a node is denoted by $\alpha[A_\alpha, \pi_\alpha]$. At this node of BBT we consider $\mathcal{S}_n^A$ as set of feasible embeddings and find a lower bound for

$$\min_{\pi \in \mathcal{S}_n^A} CAP(T, G, \pi) \tag{7.17}$$

as described in the previous subsection. (7.17) is called partial problem corresponding to the respective node. Computational experiments show that, for most of the BBT nodes, the quality of the lower bound B, computed as described in the previous subsection, is better if the characteristic sets of all BBT nodes have the following property

$$(\,(i \in A)\ \wedge\ (l(j) < l(i)\,)\quad\Rightarrow\quad(j \in A) \qquad (***)$$

Remark that this property is stronger than that described by $(**)$.

Initial permutation

In order to get an initial incumbent value for Z_{opt}, we experienced two approaches. Both of them are in principal construction procedures. At first we somehow assign two centers to two nodes of the given tree G; this is our initial partial embedding. Then in each step of the algorithm one of the remaining centers is located at one of the nodes which are still free according to some specified criteria. The resulting embedding at the end of the procedure is the initial π_{opt} and $Z_{opt} := CAP(T, G, \pi_{opt})$.

The first approach is a GRASP like heuristic (see [58]). At first, a maximum entry of the communication matrix, say $t_{i_0 j_0}$, is randomly chosen. Then, an edge (k_0, l_0) of

the given tree G is randomly chosen. The centers i_0, j_0 are located at the nodes k_0, l_0, respectively. Those vertices of G at which no communication center is located yet are called *free vertices*, whereas the other vertices are called *occupied vertices*. Let us denote the current partial embedding by π_0 and the set of currently occupied vertices by A.
Next, for each pair (p, q), where p is a not yet located center and q a free node of T, the so called *partial maximum noise* B_{pq} is computed:

$$B_{pq} = \max_{1 \le i \le n} B_{\pi_{pq}}(i) \ ,$$

where π_{pq} is the partial embedding which locates the centers of A at the same vertices of G as π_0 does and, additionally, locates q at p. ($B_{\pi_{pq}}(i)$ is defined as in (7.12) corresponding to π_{pq} and $A \cup \{p\}$ instead of π_0 and A, respectively.) Then, a pair (p_0, q_0) is chosen randomly such that $B_{p_0 q_0} = \min\limits_{p,q} B_{pq}$. The current permutation π_0 and the set A are updated by setting $\pi_0(p_0) = q_0$ and $A = A \cup \{p_0\}$, respectively. This step is repeated until all the centers have been located. At each step besides the first one $B_{\pi_{pq}}(i)$, $1 \le i \le n$, is computed as:

$$B_{\pi_{pq}}(i) = B_{\pi_0}(i) + \sum_{j \in A_i^*} \left(t_{\pi_0(j),q} + t_{q,\pi_0(j)} \right) \ ,$$

where $A_i^* = \{j \in A : i \in P(j,p)\}$. This procedure is rather time consuming; it takes $O(n^5)$ time.

The second approach is again a greedy like heuristic that uses the length d_{ij} of the path $P(i,j)$ joining i and j, for any pair (i,j) of vertices in the given tree. Exploiting the rooted structure of the tree, d_{ij}, $1 \le i,j \le n$, can be computed in $O(n^2)$ time. This time, trying to improve the time complexity of the procedure, we do not compute the partial values of the objective function at each iteration.

At first, a pair (k_0, l_0) such that $t_{k_0 l_0}$ minimizes t_{kl}, $1 \le k,l \le n$, is randomly chosen. Then, a pair (i_0, j_0) such that $d_{i_0 j_0} = \max_{1 \le i,j \le n} d_{ij}$ is randomly chosen. The centers k_0, l_0 are located at i_0, j_0, respectively. In this way the initial partial embedding π_0 is generated. Let us denote by A the set of the currently occupied vertices of G. Next, among the pairs of centers (p, q), such that at least one of p, q is not yet located, a pair (p_0, q_0) which minimizes t_{pq} is randomly chosen. There are two cases: Either one of these centers, say p_0, is already located at some vertex, say u, or none of p_0, q_0 is located yet. In the first case, center q_0 is located at a vertex k_0 randomly chosen among those vertices which maximize the distance d_{uk} over all free nodes k of G. The partial embedding π_0 and set A are updated by setting $\pi_0(k_0) = q_0$ and $A := A \cup \{k_0\}$, respectively. In the second case a pair (k_0, l_0) is randomly chosen among those pairs of vertices which maximize the distance d_{kl} over all pairs (k, l) of free vertices in T. p_0 and q_0 are then located at k_0 and l_0, respectively. The partial embedding π_0 and set A are updated by setting $\pi_0(k_0) = p_0$, $\pi_0(l_0) = q_0$ and $A := A \cup \{k_0, l_0\}$, respectively. This step is repeated until all centers have been located. It is easily seen that the time complexity of this algorithm is $O(n^3)$.

Fathoming and branching

At each node of the BBT we calculate a lower bound as described in the previous subsection. If this lower bound is greater or equal to Z_{opt} we fathom all branches emanating from this node, otherwise we branch at this node as described in the next paragraph. In case of branching at node α, we derive an upper bound equal to $CAP(T, G, \pi_1)$ for the optimal solution of the partial problem associated to node α, where $\pi_1 \in \mathcal{S}_n^{A_\alpha}$. The embedding π_1 is generated by applying the second approach described above with π_α as initial partial embedding and with A_α as initial set A of occupied nodes.
$CAP(T, G, \pi_1)$ is called *upper bound at the corresponding node.* If $CAP(T, G, \pi_1) < Z_{opt}$ we set $Z_{opt} := CAP(T, G, \pi_1)$ and $\pi_{opt} := \pi_1$.

Branching policy and priority rules

Let us suppose that we are branching at node $\alpha[A_\alpha, \pi_\alpha]$ of BBT. As described at the beginning of this subsection, this node is characterized by the subset A_α of the occupied vertices and by the partial embedding π_α, which assigns centers to the vertices of A_α.
Our branching policy cares that, for each node α, the corresponding set A_α has property $(***)$. In this way we can apply Observation 7.4 for the vertices of A_α.
For branching at node α we select a vertex $i_0 \notin A_\alpha$ such that $l(i_0) = \min\{l(j): j \notin A_\alpha\}$. Moreover, in case of depth first search (DFS) we choose the vertex i_0 such that:

$$\max_{k\in A} d_{i_0 k} = \max\left\{\max_{k\in A} d_{ik}: i \notin A_\alpha\, , l(i) = \min_{j\notin A_\alpha} l(j)\right\}.$$

Another possible choice for i_0 is described by:

$$\max_{k\in A} d_{i_0 k} = \min\left\{\max_{k\in A} d_{ik}: i \notin A_\alpha\, , l(i) = \min_{j\notin A_\alpha} l(j)\right\}.$$

Let us denote $\gamma := n - |A_\alpha|$ and let us branch from α to the nodes α_1, α_2, ..., α_γ. The characteristic sets and partial embeddings of these nodes are given by the following equalities:

$$A_{\alpha_j} = A_\alpha \cup \{i_0\}, \qquad j = 1, 2, \ldots, \gamma \tag{7.18}$$

$$\pi_{\alpha_j}(i) = \begin{cases} \pi_\alpha(i) & i \in A_\alpha \\ k_j & i = i_0 \end{cases} \tag{7.19}$$

where $\{1, 2, \ldots, n\} \setminus A_\alpha = \{k_1, k_2, \ldots, k_\gamma\}$. In case of DFS the order of the elements k_i is important. They are ordered such that $\max\limits_{h\in A_\alpha}\{t_{k_i \pi_\alpha(h)}\}$ does not increase when i increases. Figure 7.4 illustrates the branching at node α of the BBT.

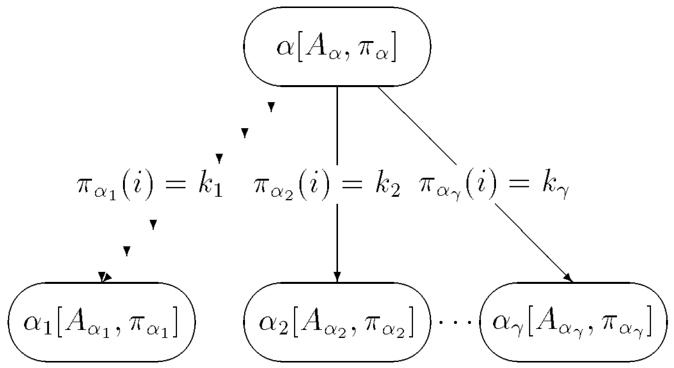

$\alpha[A_\alpha, \pi_\alpha]$
$\pi_{\alpha_1}(i) = k_1$
$\pi_{\alpha_2}(i) = k_2$
$\pi_{\alpha_\gamma}(i) = k_\gamma$
$\alpha_1[A_{\alpha_1}, \pi_{\alpha_1}]$
$\alpha_2[A_{\alpha_2}, \pi_{\alpha_2}]$
$\cdots$
$\alpha_\gamma[A_{\alpha_\gamma}, \pi_{\alpha_\gamma}]$

Figure 7.4: Branching at the node α of BBT

We have experimented with the following branching rules.

(i) depth first search (DFS).

(ii) branching at the node with the smallest lower bound.

(iii) branching at the node with the smallest upper bound.

(iiii) branching at the node with the smallest arithmetic mean of the lower and upper bound.

Among the four tested branching policies, DFS gives the best results. In the following subsection we show that in this case the time for computing lower bounds can be reduced at most of BBT nodes.

7.2.4 Speeding up the lower bound calculation

At BBT nodes of level 1, i.e. sons of the BBT root, the respective lower bounds are computed as described in Subsection 7.2.2. For the other nodes the lower bound computation can be speeded up as shown below. At each node of the BBT, say $\alpha[A_{\alpha,\pi_\alpha}]$, we save the active indices for all vertices of G, the Ω-vector of T with respect to A_α and π_α and a vector V_α of size n defined as follows:

$$V_\alpha(i) = \begin{cases} N_{\pi_\alpha}(i) & \text{if} \quad i \in A_\alpha \\ B^{(1)}_{\pi_\alpha}(i) & \text{if} \quad i \notin A_\alpha, R^-(i) \subseteq A_\alpha \\ B_{\pi_\alpha}(i) & \text{if} \quad i \notin A_\alpha, R^-(i) \not\subseteq A_\alpha \end{cases} \tag{7.20}$$

The recursive computation of this vector takes $O(n^2)$ steps per BBT node, whereas the computation of the active index for a given vertex may be done in $O(n)$ time and the Ω-vector computation takes $O(n^2)$ time.
Indeed, at each node α of BBT and for each i such that $i \notin A_\alpha$, but $R^-(i) \subseteq A_\alpha$ we save a vector $W_\alpha^{(i)}$ of size $n+1$ defined as follows:

$$W_\alpha^{(i)}(k) = \begin{cases} M & \text{if} \quad k \in \pi_\alpha(R^-(i)) \\ \sum\limits_{j \in R^-(i)} \left(t_{k\pi_\alpha(j)} + t_{\pi_\alpha(j)k}\right) & \text{if} \quad k \notin \pi_\alpha(R^-(i)) \\ \sum\limits_{l \notin \pi_\alpha(R^-(i))} W_\alpha^{(i)}(l) & \text{if} \quad i = n+1 \end{cases} \tag{7.21}$$

where M is a large positive number (larger than the sum of all entries of the matrix T). These vectors are also computed recursively in $O(n^2)$ time per node of BBT, as shown below.
Suppose we are branching at the node α of BBT as shown in Figure 7.4. Let us denote by i the vertex to be added to A_α at the current branching step. Assume that we are computing the lower bound at the node α_1. (The same arguments hold for the other nodes α_k, too.) For each vertex j of T the lower bound of the noise at this vertex can be computed in $O(n)$ time. Indeed, let us distinguish four cases:

1. $j \in A_\alpha$,

2. $j = i$

3. $j \notin A_{\alpha_1}$ and $R^-(j) \subseteq A_{\alpha_1}$

4. $j \notin A_{\alpha_1}$ and $R^-(j) \not\subset A_{\alpha_1}$

1. If $j \in A_\alpha \subset A_{\alpha_1}$, $N_\pi(j) = N_{\pi_\alpha}(j) = V_\alpha(j)$, $\forall \pi \in S_n^{A_{\alpha_1}} \subset S_n^{A_\alpha}$ (See Observation 7.4). Therefore, there is no need of any computational efforts for calculating the noises at such vertices.
2. If $j = i$, the noise $N_{\pi_{\alpha_1}}(i) = N_\pi(i)$, $\forall \pi \in S_n^{\pi_{\alpha_1}}$ (See Observation 7.4) can be computed in $O(n^2)$ time.
3. In the case that $j \notin A_{\alpha_1}$ and $R^-(j) \subseteq A_{\alpha_1}$, either $R^-(j) \subseteq A_\alpha$ or j is the father of i. If j is the father of i, $B^*_{\pi_{\alpha_1}}(j)$ can be computed as in Lemma 7.5. This takes $O(n^2)$ time.
In the other case, namely $R^-(j) \subseteq A_\alpha$, we have

$$V_{\alpha_1}(j) = B^{(1)}_{\pi_{\alpha_1}}(j) = B^{(1)}_{\pi_\alpha}(j) = V_\alpha(j).$$

Then, $B^*_{\pi_{\alpha_1}}(j)$ is given as follows:

$$B^*_{\pi_{\alpha_1}}(j) = B^{(1)}_{\pi_\alpha}(j) + C^{(j)}_{\alpha_1}(n+1) - \max_{k \notin \pi_{\alpha_1}(A_{\alpha_1})} C^{(j)}_{\alpha_1}(k)$$

Considering that the vectors $C_\alpha^{(j)}$ and $C_{\alpha_1}^{(j)}$ are identical, $B^*_{\pi_{\alpha_1}}(j)$ can be computed in $O(n)$ time, for j as above.
4. Let $j \notin A_{\alpha_1}$ and $R(j) \not\subset A_{\alpha_1}$. For a fixed vertex j, either $i \in R(j)$ or $i \notin R(j)$. In the case that $i \in R(j)$, moving from i to j a son t of j can be found such that $i \in R(t)$. Then, $B_{\pi_{\alpha_1}}(j)$ is computed as follows:

$$\begin{aligned} B_{\pi_{\alpha_1}}(j) &= B_{\pi_\alpha}(j) + \sum_{\substack{l \in S(j) \\ l \neq t}} \sum_{k \in R_{A_\alpha}(l)} \left(t_{\pi_{\alpha_1}(i)\pi_{\alpha_1}(k)} + t_{\pi_{\alpha_1}(k)\pi_{\alpha_1}(i)} \right) \\ &\quad + \sum_{k \in A_{\alpha_1} \setminus R(j)} \left(t_{\pi_{\alpha_1}(i)\pi_{\alpha_1}(k)} + t_{\pi_{\alpha_1}(k)\pi_{\alpha_1}(i)} \right) \end{aligned}$$

The last computation trivially takes $O(n)$ time. In case that $i \notin R(j)$, $B_{\pi_{\alpha_1}}(j)$ is calculated as follows:

$$B_{\pi_{\alpha_1}}(j) = B_{\pi_\alpha}(j) + \sum_{k \in R_{A_\alpha}(j)} \left(t_{\pi_{\alpha_1}(k)\pi_{\alpha_1}(i)} + t_{\pi_{\alpha_1}(i)\pi_{\alpha_1}(k)} \right)$$

This computation also takes $O(n)$ time. Clearly, in the same time we have also computed $W_{\alpha_1}(j)$, for all $j \notin A_{\alpha_1}$, $R^-(j) \in A_{\alpha_1}$. In order to derive $B'_{\pi_{\alpha_1}}(j)$ we still need $B^+_{\pi_{\alpha_1}}(j)$. Given the respective Ω-vector and the active index of the node j, only $O(n)$ time is needed to compute the additive bound $B^+_{\pi_{\alpha_1}}(j)$. Thus, the following theorem holds:

Theorem 7.6 *In case of DFS, the lower bounds described in Subsection 7.2.2 can be computed in $O(n^2)$ time at each node of BBT, besides those of the first level.* ∎

Obviously, this profit in time "costs" some memory. Namely, we need memory to save the vectors V_α, $W_\alpha^{(i)}$, the active indices of the vertices of T and the Ω-vector at each node of BBT. Summarizing, this amounts to $O(n^2)$ per node of BBT. Considering that DFS is applied, there is no need of saving these variables for more than n BBT nodes simultaneously, where n is the size of the problem. Thus, we have reduced the time complexity of the lower bounds calculation at costs of $O(n^3)$ memory.

7.2.5 Numerical results and comments

The branch and bound algorithm described in the previous subsection is implemented as a C code on a DIGITAL DECstation 5000/240. The algorithm has been tested for random trees having 7 to 15 vertices. For trees with more vertices the BBT increases enormously. The rates of message exchange are random numbers from 1 up to 100.
MINMAX1 and MINMAX2 are respectively the versions of the algorithm corresponding to the first and the second approach for generating the initial embedding, both of them using DFS. Table 7.1 shows the performance of MINMAX1 and MINMAX2. The results corresponding to each problem size are averages of data we got for 10 test

runnings of the related algorithm on instances of the corresponding size. In the first and fourth columns of the table the relative errors of the initial solution with respect to the optimal one are presented for MINMAX1 and MINMAX2, respectively. In the second and fifth column, the corresponding amount of branched BBT nodes is shown, whereas the third and sixth column represent the nodes at which an optimal solution was found for MINMAX1 and MINMAX2, respectively. It is worthy to remark that in all our test runnings the optimal solution was found relatively early, namely, within the first half of the branched BBT nodes.

In 80% of the cases not more than one fifth of the branched nodes were needed to find the optimal solution. The performance of both algorithms depends strongly on the quality of the initial solution, as can be seen in Table 7.1. As the running times for problems of size larger than 10 exceed 1 hour, we conclude that it may be a fruitful approach to first derive good heuristics and then probably to use them for generating a better initial solution for the branch and bound algorithm. Due to the earliness of finding the optimal solution, one might get good suboptimal solutions by applying time limits to the branch and bound code. For the instances we tested 30 minutes seems to be a reasonable time limit.

Table 7.1: The performance of MINMAX1 and MINMAX2

size	**MINMAX1**			**MINMAX2**		
	init. rel. error (%)	branched nodes	opt. sol. at node	init. rel. error (%)	branched nodes	opt. sol. at node
7	19.5	353	149	17.9	304	86
8	28.6	3339	1896	18.1	3241	1508
9	21.9	35632	5237	24.6	35615	5306
10	22.4	124855	52789	22.8	124800	52765
11	20.2	134966	47323	16.5	125877	42378
12	21.4	187356	57862	18.4	175498	52345
13	20.7	201456	67321	19.6	195647	62765
14	20.9	240567	73456	20.2	236748	72895
15	21.3	289871	85769	19.5	258975	79874

In most of the cases MINMAX2 yields better results than MINMAX1. The better quality of the initial solutions used by MINMAX2 with respect to those used by MINMAX1 causes the decrease of the amount of branched BBT nodes when MINMAX2 is applied.
Computational results show that in the case of embedding into a path, the quality of the initial solution generated by applying the second approach is generally much

better than in the case of embedding into an arbitrary tree. We observed also that, generally, the smaller the maximal degree of the vertices of the given tree G the better is the performance of both algorithms. Again, the better performance is due to better initial solutions derived as described in Subsection 7.2.3.

7.3 Complexity results

In this section computational complexity aspects of the CAP are discussed, aiming to single out polynomially solvable versions of the problem in the case that the underlying graph has a simple structure. It will turn out that in most of cases the problem remains NP-hard. However, two polynomially solvable versions of the problem are described. In the NP-completeness proofs, reductions from the following NP-complete problems (cf. Garey and Johnson [68]) are used:

Partitioning (Part)

Input: A set of m positive integers $a_1, \ldots, a_m$, $\sum_{i=1}^m a_i = 2B$.

Question: Does there exist a subset $I \subseteq \{1, \ldots, m\}$ with $\sum_{i\in I} a_i = \sum_{i\notin I} a_i$=B ?

Partition into Triangles (PΔ)

Input: A graph $G_1 = (V_1, E_1)$ with $|V_1| = 3q$ and $|E_1| \geq 3|V_1|$.

Question: Does there exist a partition of V_1 into q triples such that every triple spans a triangle in G_1 ?

Remark. The density restriction $|E_1| \geq 3|V_1|$ does not appear in the standard formulation of PΔ in [68]. However, by adding a large disjoint clique to the input graph, we can make the input graph arbitrarily dense without changing the solvability of the problem. Hence, PΔ is NP-complete also for dense graphs with $|E_1| \geq 3|V_1|$.

7.3.1 Paths and cycles

Consider an instance of problem PART and let $V = \{v_1, v_2, \ldots, v_{m+3}\}$ and $E = \{(v_i, v_{i+1})\colon 1 \leq i \leq m+2\}$ define a linear network (path). Consider the following traffic matrix for $m+3$ communication centers. The only transmitting center is C_{m+3}, all other centers are just receiving messages. C_{m+3} sends $2B+2$ messages to C_{m+1}, $2B+2$ messages to C_{m+2} and a_i messages to C_i, $1 \leq i \leq m$. We claim that there exists an embedding π with

$$\max\{N_\pi(i)\colon 1 \leq i \leq n\} \leq B$$

if and only if the above instance of PART is solved by "yes".

(If) If the instance of PART has a solution I with $\sum_{i \in I} a_i = B$, let $\pi_0 \in \mathcal{S}_{m+3}$ embed the centers C_i, $i \in I$, arbitrarily at the vertices $v_1, v_2, \ldots, v_{|I|}$ and the centers C_i, $i \in \{1 \ldots m\} - I$, arbitrarily at the vertices $v_{|I|+4}, \ldots, v_{m+3}$. Centers C_{m+1}, C_{m+3}, C_{m+2} are assigned (in this ordering) to the vertices $v_{|I|+1}$, $v_{|I|+2}$ and $v_{|I|+3}$. It is straight forward to check that $N_{\pi_0}(|I|+1) = N_{\pi_0}(|I|+3) = B$ and that smaller noises correspond to all other vertices.

(Only if) Conversely, assume that there exists an embedding π_0 with

$$\max\{N_{\pi_0}(i) \colon 1 \leq i \leq m+3\} \leq B\,.$$

Clearly, the centers C_{m+1} and C_{m+2} must be embedded immediately to the right and immediately to the left of C_{m+3} (the noise corresponding to any other center in between would be at least $2B+2$). Center C_{m+3} sends an overall amount of $2B$ messages to the remaining centers and all these messages must cross either C_{m+1} or C_{m+2}. The claim follows.

A similar construction works for a cyclic communication network on $m+6$ vertices. The above traffic matrix is extended by three more centers C_{m+4}, C_{m+5} and C_{m+6}. C_{m+6} sends $2B+2$ messages to each center C_{m+4} and C_{m+5}. C_{m+4} sends B messages to C_{m+5}. These three new centers neither send nor receive messages from other centers. It is easy to see that they must be embedded on three adjacent vertices of the cycle and thus leave a path on $m+3$ vertices for the remaining vertices. Summarizing, we formulate the following two theorems.

Theorem 7.7 *Finding an embedding of a communication network into a path to minimize the maximum intermediate traffic is NP-complete.* ■

Theorem 7.8 *Finding an embedding of a communication network into a cycle to minimize the maximum intermediate traffic is NP-complete (in the single path model and in the fractional model).* ■

7.3.2 Stars of bounded branch length

A star consists of a central vertex and several paths that are connected to this central vertex. The *branch length* of a star is the number of edges of the longest path connected to the central vertex. In this subsection, we derive two results for the CAP on stars of bounded branch length. First, we state an NP-completeness result for the CAP on stars of branch length three. Secondly, we derive a polynomial time algorithm for the problem on stars of branch length two.

Theorem 7.9 *Finding an embedding of a communication network into a star of branch length three to minimize the maximum intermediate traffic is NP-complete.*

Proof. Consider an instance of problem PΔ as defined above. Construct a traffic matrix with $3q+4$ communication centers that are to be embedded into the following graph G. G consists of a central vertex and of $q+1$ branches of length three that leave the central vertex. For every vertex v_i in V_1 (the vertex set of the graph in PΔ) there is a corresponding communication center C_i. For $1 \leq i,j \leq 3q$, the amount of traffic between C_i and C_j is defined symmetrically $t_{ij} = t_{ji} = 1$ in the case that there is an edge $(v_i, v_j) \in E_1$ and $t_{ij} = t_{ji} = 0$ in case there is no edge in E_1 between these vertices. Moreover, there are four communication centers C_{3q+1}, C_{3q+2}, C_{3q+3}, C_{3q+4} that do neither send nor receive messages from the communication centers C_i with $i \leq 3q$. The traffic among these four centers is defined by $t_{3q+1\,3q+2} = t_{3q+2\,3q+3} = t_{3q+3\,3q+4} = X+1$ with $X = 2|E_1| - 2|V_1|$ and by $t_{ij} = 0$ for all other $3q+1 \leq i,j \leq 3q+4$.

We claim that for the traffic matrix $T = (t_{ij})$ there exists an embedding π_0 into the star G with maximum noise $CAP(T, G, \pi_0) = X = 2|E_1| - 2|V_1| \geq 4|V_1|$ if and only if the graph $G_1 = (V_1, E_1)$ has a partition into q triangles.

(If) Suppose, a partition into triangles exists. Let $\pi_0 \in \mathcal{S}_{3q+4}$ embed the centers corresponding to the vertices of every triangle in arbitrary order in one of the branches of the network. Let π_0 embed center C_{3q+1} to the central vertex. Moreover, π_0 embeds centers C_{3q+2}, C_{3q+3}, C_{3q+4} on the remaining branch in this order. In this embedding C_{3q+2}, C_{3q+3} and C_{3q+4} do not suffer from any intermediate traffic. C_{3q+1} is crossed by messages that correspond to edges that do not belong to the triangles in the partition. Since there are $|V_1|$ edges (corresponding to $2|V_1|$ messages) in the triangles, C_{3q+1} is crossed by X messages. Finally, consider three centers C_i, C_j, C_k that correspond to vertices in V_1 and that are embedded together on a branch. Assume that C_i is embedded next to the central vertex and thus all messages going to and coming from C_k and C_j cross C_i. In the worst case, C_k and C_j communicate with all other $|V_1|$ vertices and $4|V_1| \leq X$ intermediate traffic crosses C_i. Thus, the maximum of the noises $N_{\pi_0}(i)$, $i \leq 1 \leq 3q+4$, equals X.

(Only if) Now suppose that there exists an embedding π_0 with $CAP(T, G, \pi_0) \leq X$. It is easy to see that the communication centers C_{3q+1}, C_{3q+2}, C_{3q+3}, and C_{3q+4} occupy the central vertex and one of the branches. We claim that the embedding of the other centers into the remaining branches induces a partition into triangles of the corresponding vertices. Otherwise, the noise at the central vertex would result from at least $|E_1| - |V_1| + 1$ edges that amounts to $2(|E_1| - |V_1| + 1) = X + 2$ messages. ■

Theorem 7.10 *Finding an embedding of a communication network into a star of branch length two to minimize the maximum intermediate traffic can be done in* $O((\log(n) + \log(M)) \cdot n^4)$ *time, where* $M = \max_{1 \leq i,j \leq n} t_{ij}$.

Proof. Consider a star $G = (V, E)$ of branch length 2 on n vertices with n_1 branches of length 1 and n_2 branches of length 2 ($n = n_1 + 2n_2 + 1$). Let $C_1, \ldots, C_n$ denote the communication centers with traffic t_{ij} being sent from C_i to C_j. In a first step, we investigate the following special case of the problem: Given a number $K \geq 0$ and a center C^*, does there exist an embedding of the centers and a routing of the messages such that C^* is embedded into the central vertex and such that the resulting maximum noise is at most K ?

W.l.o.g. assume that $C_n = C^*$ has to be embedded into the central vertex. We construct an edge weighted undirected complete graph $(U, U \times U)$ on $2n_1 + 2n_2$ vertices $u_1, \ldots, u_{|U|}$ as follows. For $1 \leq i \leq n-1$, vertex u_i corresponds to communication center C_i. Vertices u_i with $n \leq i \leq 2n_1 + 2n_2$ are so called dummy vertices. For $1 \leq i, j \leq n-1$, define by

$$\textsc{OutTwo}(i,j) = \sum_{k=1}^{n-1}(t_{ik} + t_{jk}) - (t_{ij} + t_{ji})$$

the *out-traffic* of the pair C_i and C_j, i.e. the overall amount of traffic moving from $\{C_i, C_j\}$ to $\mathcal{C} - \{C_i, C_j, C_n\}$. For $1 \leq i, j \leq n$, we denote by

$$\textsc{OutOne}(i,j) = \sum_{k=1}^{n}(t_{ik} + t_{ki}) - (t_{ij} + t_{ji})$$

the overall amount of traffic moving between $\{C_i\}$ and $\mathcal{C} - \{C_i, C_j\}$. The weights of edges between dummy vertices are set to $K+1$. The weight of an edge between a vertex u_i with $1 \leq i \leq n-1$ and a dummy vertex is $\textsc{OutOne}(i, n)$. The weight of an edge between two vertices u_i and u_j with $1 \leq i, j \leq n-1$ is defined as

$$\begin{array}{cc} K+1, & \text{if } \min\{\textsc{OutOne}(i,j), \textsc{OutOne}(j,i)\} \geq K \\ \textsc{OutTwo}(i,j), & \text{otherwise} \end{array}$$

We claim that the graph $(U, U \times U)$ has a perfect matching of weight at most K if and only if there exists an embedding of the communication centers into the star (V, E) with corresponding maximum noise at most K, where C_n is embedded into the central vertex.

(If) Assume that there is a perfect matching with weight K and consider the following embedding. The n_1 communication centers whose corresponding vertices are matched with dummy vertices are embedded into the n_1 branches of length one. Every pair of communication centers whose corresponding vertices are matched with each other is embedded into one of the n_2 branches of length two. There are two possible embeddings for such a pair; we always choose that embedding that locally minimizes the intermediate traffic. (If C_i is embedded at the middle vertex of a branch and C_j at the end vertex, then noise at the vertex where C_i is embedded equals $\textsc{OutOne}(j, i)$ and the noise at the vertex where C_j is embedded equals 0. Depending on $\textsc{OutOne}(i, j)$ and $\textsc{OutOne}(j, i)$ we can locally minimize the maximum of the noise at these two vertices.) Let us denote such an embedding by π_0.

What is the maximum noise corresponding to π_0? For vertices at the ends of the branches, there is no intermediate traffic. For a vertex v in the middle of a length two branch let C_j be embedded at the end of the branch and $\pi_0(v) = i$. Then,

$$N_{\pi_0}(v) = \textsc{OutOne}(j, i) \leq K$$

holds. Indeed, otherwise we would have

$$\textsc{OutOne}(j, i) = \min\{\textsc{OutOne}(i, j), \textsc{OutOne}(j, i)\} > K\,,$$

which implies the weight of the edge (u_i, u_j) to be equal to $K + 1$. This contradicts that fact that (u_i, u_j) is an edge of the matching.
Finally, the noise at the central vertex (where center $C_n = C^*$ is embedded) is given as the sum of all messages going from one branch to another. A center C_i that is embedded in a length one branch causes noise $\textsc{OutOne}(i, n)$ and two centers C_i and C_j embedded in a length two branch cause noise $\textsc{OutTwo}(i, j)$. Hence, the noise at the central vertex equals the weight of the matching.

(Only if) Now assume that there is an embedding π_0 which embeds C_n to the central vertex and whose maximum corresponding noise is at most K. Consider the matching in $(U, U \times U)$ that results from matching every center on a length one branch with a dummy vertex and from matching every pair of centers on a length two branch with each other. The overall weight of this matching equals the maximum noise corresponding to π_0 which is at most K.

A minimum weight perfect matching in the complete graph $(U, U \times U)$ can be computed in $O(|U|^3)$ time. Therefore, the special case with a given number $K \geq 0$ and a fixed center C^* can be solved in $O(n^3)$ time, since $|U| = O(n)$. For any fixed central communication center C^* an embedding which places C^* at the central vertices and yields the smallest possible maximum maximum noise can be computed in $O((\log(n)+\log(M))\cdot n^3)$ time by applying binary search. (M is as specified above.) Repeating this procedure for every communication center embedded into the central vertex completes the proof. ■

7.3.3 Doublestars

In this subsection, we investigate networks that are *doublestars*. Recall that a doublestar $K_{2,m}$ consists of two central vertices that are connected to every other vertex (i.e., a complete bipartite graph where one class of the bipartition consists of two vertices). For doublestars, the complexity of CAP depends on the model. In the single path model, the problem is NP-complete. In the fractional model, the problem is equivalent to solving a polynomial number of linear programs and thus it can be solved in polynomial time.

Theorem 7.11 *Finding an embedding of a communication network into a doublestar to minimize the intermediate traffic is NP-complete in the single path model.*

Proof. Consider an instance of problem PART, i.e. m positive integers $a_1, \dots, a_m$ with $\sum_{i=1}^{m} a_i = 2B$ and $2m+4$ communication centers. We construct a traffic matrix as follows. For $1 \leq i \leq m$, $t_{i\,i+m} = a_i$. Moreover,

$$t_{2m+1\,2m+2} = t_{2m+2\,2m+3} = t_{2m+3\,2m+4} = t_{2m+4\,2m+1} = B+1.$$

All other values t_{ij} are zero. We claim that these centers have an embedding into the doublestar $K_{2,2m+2}$ with maximum intermediate traffic at most B if and only if the instance of the partioning problem is solved by "yes".

(If) Assume that for $I \subseteq \{1 \dots m\}$, $\sum_{i \in I} a_i = B$ holds. Assign C_{2m+1}, C_{2m+3} to the two central vertices and C_{2m+2}, C_{2m+4} to two arbitrary non central vertices. Arbitrarily assign the remaining centers to the remaining vertices. It remains to specify the paths along which the messages $t_{i\,i+m}$ are sent from C_i to C_{i+m}. If $i \in I$, these messages are routed via the first central vertex and if $i \notin I$, the messages are routed via the second central vertex. The only vertices that suffer from some noise are the two central vertices; the noise at both of them equals B.

(Only if) Assume that there exist an embedding and routing with maximum corresponding noise at most B. The four centers C_{2m+1}, C_{2m+2}, C_{2m+3}, C_{2m+4} form a cycle of length four that must occupy both central vertices and two non central vertices. All other messages must be routed via one of the central vertices. Since the overall sum of noises is $2B$, the maximum noise is at most B only if a subset of the C_i, $1 \leq i \leq m$, send messages with overall size exactly B. ∎

Next, we give an equivalent formulation for the general version of the CAP in the fractional model. This formulation consists of a set a of linear programs as described below.

Suppose that the communication centers have to be embedded into an arbitrary undirected graph $G = (V, E)$. Let us denote by $\mathcal{P} = \{P_k\}$ the set of the directed paths in G and by Φ a set of representatives for all combinatorially equivalent embeddings of the centers into the network. (Two embeddings π_1 and π_2 are called *combinatorially equivalent* if the two labeled graphs which result from labeling every vertex of G with the corresponding embedded center are isomorphic.) Here "directed path" means a path whose endvertices are considered as an ordered pair. Practically, this means that each path, say connecting i and j, is considered twice, once as path from i to j and once as path from j to i. Recall that $conn(i,j)$ is the set of all paths from i to j. Consider a fixed embedding $\pi \in \Phi$ of the centers into the vertices of the graph. We can derive the routing pattern ρ which minimizes the maximum intermediate traffic over all centers C_i

$$\min_{\rho} \max\{N_{\pi,\rho}(i) \mid 1 \leq i \leq n\}$$

by solving the following linear program. (Recall that $i \in P_k$ means that i is an inner vertex of P_k.)

$$\begin{aligned}
&\min z \\
&\text{s.t.} \quad \sum_{k\,:\,P_k \in conn(i,j)} x_k = t_{\pi(i)\pi(j)} \qquad \text{for } 1 \le i,j \le n \\
&\qquad \sum_{k: i \in P_k} x_k \le z \qquad\qquad\qquad \text{for } 1 \le i \le n
\end{aligned}$$

The variables x_k state how much traffic is sent through the path P_k. The first group of restrictions guarantees that all messages are sent, whereas the second group of restrictions bounds the maximum intermediate traffic by z. This linear program has $|\mathcal{P}|$ variables and $n^2 + n$ restrictions.

For each fixed embedding $\pi \in \Phi$ we solve the corresponding linear program. Among all optimal solutions the one which yields the minimum objective function value is the optimal solution to the initial problem. Summarizing, we have the following theorem.

Theorem 7.12 *In the fractional model, finding an embedding of the communication centers $C_1, C_2, \ldots, C_n$ into an undirected network $G = (V, E)$ and a routing pattern which minimize the maximum intermediate traffic over all centers C_i, $1 \le i \le n$, can be done by solving $|\Phi|$ linear programs. Each of these linear programs has $|\mathcal{P}|$ variables and $O(n^2)$ restrictions.* ■

As corollary of the last theorem we get:

Corollary 7.13 *In the fractional model, finding an embedding of a communication network into a doublestar to minimize the maximum intermediate traffic is solvable in polynomial time.*

Proof. It is easily seen that (i) the number of paths in a doublestar with n vertices is $O(n^2)$ and that (ii) the number of equivalence classes with respect to the combinatorial equivalence of embeddings is also $O(n^2)$. Hence, Theorem 7.12 implies that in this case the optimal embedding and the optimal routing pattern can be found by solving a polynomial number of linear programs, where each linear program is of polynomial size. ■

7.4 The asymptotic behavior of the CAP on trees

In this section the asymptotic behavior of the CAP on trees is discussed. This problem shows the same asymptotical behavior as the QAP and the BiQAP under similar probabilistic assumptions.

A sequence of problems $CAP(T^{(n)}, G^{(n)})$ is considered, where $T^{(n)} = \left(t_{ij}^{(n)}\right)$ is an $n \times n$ communication matrix corresponding to n communicating centers, which should be embedded into the vertices of a tree $G^{(n)} = \left(V^{(n)}, E^{(n)}\right)$, $|V^{(n)}| = n$. We assume that the entries $t_{ij}^{(n)}$, $n \in \mathbb{N}$, $1 \le i, j \le n$, are identically distributed random variables on $[0, M]$, where M is some positive constant. Moreover, for all $n \in \mathbb{N}$, the variables $t_{ij}^{(n)}$, $1 \le i, j \le n$, are pairwise independent random variables. An optimal and a "worst" solution to $CAP(T^{(n)}, G^{(n)})$ are denoted by $\pi_{opt}^{(n)}$ and $\pi_{wor}^{(n)}$, respectively. That is:

$$\begin{aligned} CAP\left(T^{(n)}, G^{(n)}, \pi_{opt}^{(n)}\right) &= \min_{\pi \in \mathcal{S}_n} CAP(T^{(n)}, G^{(n)}, \pi) \\ CAP\left(T^{(n)}, G^{(n)}, \pi_{wor}^{(n)}\right) &= \max_{\pi \in \mathcal{S}_n} CAP(T^{(n)}, G^{(n)}, \pi) \end{aligned}$$

Moreover, for each vertex $v \in V$, where $G = (V, E)$ is a graph with vertex set V and edge set E, denote by $inn(v)$ the set of all paths which contain v as an inner vertex. $inn(v)$ is called the *active set* of v.

It can be shown that, under the above probabilistic conditions and for an arbitrary sequence of trees G_n, $n \in \mathbb{N}$, the ratio between the objective function values corresponding to an optimal solution and a worst solution of $CAP(T^{(n)}, G^{(n)})$ approaches 1 with probability tending to 1, as the size n of the problem tends to ∞. More precisely, we prove the following theorem:

Theorem 7.14 *Let $G^{(n)}$, $n \in \mathbb{N}$, be a tree on n vertices. Consider the problems $CAP(T^{(n)}, G^{(n)})$, where $T^{(n)} = \left(t_{ij}^{(n)}\right)$ is an $n \times n$ communication matrix. The entries $t_{ij}^{(n)}$, $n \in \mathbb{N}$, $1 \le i, j \le n$, are identically distributed random variables. Moreover, for each $n \in \mathbb{N}$, the variables $t_{ij}^{(n)}$, $1 \le i, j \le n$, are pairwise independently distributed. Under these conditions the following holds:*

$$\forall \epsilon > 0, \quad \lim_{n \to \infty} P\left\{ \frac{CAP\left(T^{(n)}, G^{(n)}, \pi_{wor}^{(n)}\right)}{CAP\left(T^{(n)}, G^{(n)}, \pi_{opt}^{(n)}\right)} \le 1 + \epsilon \right\} = 1\,, \tag{7.22}$$

where $\pi_{opt}^{(n)}$ and $\pi_{wor}^{(n)}$ are defined as above.

The idea of the proof is simple and consists of two steps. First, recall that the noise at a vertex v is the overall sum of traffic exchanged between centers located at such vertices that the path connecting them contains v as an inner vertex. As the traffic exchange rates are identically distributed random variables, it is intuitively clear that the noise attains its maximum at a vertex which is contained as inner vertex in a maximum number of paths, and this holds for any arbitrary embedding. Secondly, it is easy to show that the minimization of the noise at a distinguished fixed vertex of a tree can be equivalently formulated as a QAP. Then, the well known results on the asymptotic behavior of certain combinatorial optimization problems as described in Section 2.7 would apply. These ideas are mathematically formulated and proved in the lemmas 7.17 and 7.18. The following two lemmas state easy-to-prove results of technical benefit.

Lemma 7.15 *For any $n \in \mathbb{N}$, consider an arbitrary tree $G^{(n)} = (V^{(n)}, E^{(n)})$ with vertex set $V^{(n)} = \{v_1, v_2, \ldots, v_n\}$ and edge set $E^{(n)}$. For $n \in \mathbb{N}$, let $j_n \in \{1, 2, \ldots, n\}$ such that the vertex v_{j_n} has an active set of minimum positive cardinality, i.e., $|inn(v_{j_n})| = \min\{|inn(v_i)| > 0 \colon 1 \leq i \leq n\}$. Then,*

$$\lim_{n\to\infty} |inn(v_{j_n})| = \infty$$

Proof. Let us denote by d the degree of $v_{j_n} \in V^{(n)}$. From $|inn(v_{j_n})| > 0$ follows that $d \geq 2$. Let $i_1, i_2, \ldots, i_d$ be the indices of vertices adjacent to v_{j_n} in $G^{(n)}$. Deleting the edges (v_{j_n}, v_{i_k}), $1 \leq k \leq d$, produces d connected components in $G^{(n)}$, each of them containing exactly one of the vertices v_{i_k}, $1 \leq k \leq d$. Let these components have $y_1, y_2, \ldots, y_d$ vertices respectively. Obviously $\sum_{i=1}^{d} y_i = n - 1$. Moreover,

$$|inn(j_n)| \geq \sum_{\substack{1 \leq i,j \leq d \\ i \neq j}} y_i y_j$$

Now, it is elementary to check that $\sum_{i=1}^{d} y_i = n - 1$ and $d \geq 2$ imply

$$\lim_{n\to\infty} \sum_{\substack{1 \leq i,j \leq d \\ i \neq j}} y_i y_j = \infty$$ ■

Lemma 7.16 *Let $G^{(n)}$ be a tree with vertex set $V^{(n)} = \{v_1, v_2, \ldots, v_n\}$. For all $n \in \mathbb{N}$, let $v_{k_n} \in V^{(n)}$ be a vertex with an active set of maximum cardinality, i.e.,*

$$|inn(v_{k_n})| = \max\{|inn(v_i)| \colon 1 \leq i \leq n\}.$$

Then $|inn(k_n)| \in \Omega(n^2)$.

Proof. This lemma is a straightforward consequence of the following well known graph theoretical result (folklore). Consider an arbitrary tree G with vertex set V, $|V| = n$. There exists a vertex $v_i \in V$ with degree s, $s > 1$, and a partition of the set $\{v_{i_1}, v_{i_2}, \ldots, v_{i_s}\}$ of its adjacent vertices into two sets A, B ($A \cup B = \{v_{i_1}, v_{i_2}, \ldots, v_{i_s}\}$, $A \cap B = \emptyset$) with the following property. In the graph obtained by the deletion of all edges joining v_i with vertices in A, the cardinality of the connected component containing v_i is at least $n/3$. The same holds for the connected component containing v_i in the graph obtained by the deletion of all edges joining v_i with vertices in B. ■

The next lemma principally states that for independently identically distributed rates of message exchanging, the vertex where the noise reaches its maximum depends only on the structure of the tree but not on the embedding of the centers into the vertices of the tree.

Lemma 7.17 *Assume that the conditions of Theorem 7.14 are fulfilled. Then, for each $n \in \mathbb{N}$, there exist a vertex $v_{k_n} \in V^{(n)}$ with index k_n such that*

$$\forall \epsilon > 0 \ , \ \lim_{n\to\infty} P\left\{\frac{CAP(T^{(n)}, G^{(n)}, \pi_n)}{N_{\pi_n}(k_n)} \le 1+\epsilon\right\} = 1, \tag{7.23}$$

where $\pi_n \in \mathcal{S}_n$.

Proof. Let μ and σ^2 be the expected value and the variance of the entries of the communication matrices $T^{(n)}$, $n \in \mathbb{N}$, respectively. W.l.o.g., we can assume that $\mu > 0$. Otherwise, $t_{ij}^{(n)} = 0$, for $n \in \mathbb{N}$, $1 \le i,j \le n$, and the problem would become trivial. Similarly we can assume that $\sigma^2 > 0$; otherwise t_{ij}, $1 \le i,j \le n$ would be constant and $CAP(T^{(n)}, G^{(n)})$ would be trivial. (All embeddings would imply the same maximum noise.)

For each $n \in \mathbb{N}$, fix a permutation $\pi_n \in \mathcal{S}_n$. Denote by v_{k_n} a vertex in V_n, $n \in \mathbb{N}$, with active set $inn(k_n)$ of maximum cardinality, i.e.:

$$|inn(k_n)| = \max\{|inn(i)| \colon 1 \le i \le n\}\,.$$

We show that for such vertices v_{k_n} assertion (7.23) holds and this completes the proof.

Note that the following equality holds for $n \in \mathbb{N}$, $1 \le k \le n$:

$$N_{\pi_n}(k) = |inn(k)|\mu + |inn(k)|\left(\frac{\sum\limits_{(i,j):k\in P(i,j)} t_{\pi_n(i)\pi_n(j)}}{|inn(k)|} - \mu\right) \tag{7.24}$$

Let $l_n \in \{1,2,\ldots,n\}$, $n \in \mathbb{N}$, be an index where the noise implied by π_n attains its maximum. That is $N_{\pi_n}(l_n) = CAP(T^{(n)}, G^{(n)}, \pi_n)$. Using equality (7.24) for both vertices k_n and l_n we get

$$\frac{CAP(T^{(n)}, G^{(n)}, \pi_n)}{N_{\pi_n}(k_n)} = \frac{|inn(l_n)|(\mu+y)}{|inn(k_n)|(\mu+x)}$$

where

$$x = \frac{\sum\limits_{(i,j):k_n\in P(i,j)} t_{\pi_n(i)\pi_n(j)}}{|inn(k_n)|} - \mu\ , \quad y = \frac{\sum\limits_{(i,j):l_n\in P(i,j)} t_{\pi_n(i)\pi_n(j)}}{|inn(l_n)|} - \mu\,.$$

Applying Chebyshev's inequality we have:

$$\forall \epsilon > 0\ , \quad P\{|x| > \epsilon\} \le \frac{\sigma^2}{\epsilon^2|inn(k_n)|}, \qquad P\{|y| > \epsilon\} \le \frac{\sigma^2}{\epsilon^2|inn(l_n)|} \tag{7.25}$$

Now let us fix an $\epsilon > 0$ and show that assertion (7.23) holds for that ϵ. We choose $0 < \epsilon_0 < \mu$ such that $\frac{\mu+\epsilon_0}{\mu-\epsilon_0} \le 1+\epsilon$. From (7.25) and from the choice of the indices l_n, k_n we get:

$$1 \le \frac{CAP(T^{(n)}, G^{(n)}, \pi_n)}{N_{\pi_n}(k_n)} \le \frac{\mu+\epsilon_0}{\mu-\epsilon_0} \le 1+\epsilon$$

The above inequalities hold with probability larger or equal to

$$\left(1 - \frac{\sigma^2}{\epsilon_0^2 |inn(l_n)|}\right)\left(1 - \frac{\sigma^2}{\epsilon_0^2 |inn(k_n)|}\right).$$

Thus

$$P\left\{\frac{CAP(T^{(n)}, G^{(n)}, \pi_n)}{N_{\pi_n}(k_n)} \le 1+\epsilon\right\} \ge \left(1 - \frac{\sigma^2}{\epsilon_0^2 |inn(l_n)|}\right)\left(1 - \frac{\sigma^2}{\epsilon_0^2 |inn(k_n)|}\right) \quad (7.26)$$

According to Lemma 7.15 $\lim\limits_{n\to\infty} |inn(l_n)| = \infty$ and $\lim\limits_{n\to\infty} |inn(k_n)| = \infty$. Combining this fact with (7.26) completes the proof. ■

Lemma 7.17 naturally suggests to investigate the asymptotic behavior of the ratio between the maximal and the minimal noise at vertex v_{k_n}:

$$\frac{\max\{N_\pi(k_n) \colon \pi \in \mathcal{S}_n\}}{\min\{N_\pi(k_n) \colon \pi \in \mathcal{S}_n\}}, \quad (7.27)$$

where v_{k_n} is as specified in Lemma 7.17. Notice that $N_\pi(k_n)$ can be actually seen as the objective function of a QAP instance. Indeed, for each instance $CAP(T^{(n)}, G^{(n)})$, $n \in \mathbb{N}$, let us introduce a matrix $B^{(n)} = \left(b_{ij}^{(n)}\right)$, by setting $b_{ij}^{(n)} = 1$ if $k_n \in P(i,j)$ and $b_{ij}^{(n)} = 0$ otherwise. Then, the following equality holds for each $\pi \in \mathcal{S}_n$

$$N_\pi(k_n) = Z(T^{(n)}, B^{(n)}, \pi),$$

where $Z(T^{(n)}, B^{(n)}, \pi)$ is the objective function of $QAP(T^{(n)}, B^{(n)})$. It can be easily shown that for the sequence of problems $QAP(T^{(n)}, B^{(n)})$, $n \in \mathbb{N}$, the conditions of Theorem 2.12 are fulfilled. Then, by applying this theorem we get:

Lemma 7.18 *Consider the sequence of problems $CAP(T^{(n)}, G^{(n)})$, $n \in \mathbb{N}$ and assume that the conditions of Theorem 7.14 are fulfilled. The following equality holds for each $\epsilon > 0$:*

$$\lim_{n\to\infty} P\left\{\frac{\max\{N_\pi(k_n) \colon \pi \in \mathcal{S}_n\}}{\min\{N_\pi(k_n) \colon \pi \in \mathcal{S}_n\}} \le 1+\epsilon\right\} = 1, \quad (7.28)$$

where k_n is chosen such that vertex v_{k_n} has active set of maximum cardinality.

Proof. We show how to apply Theorem 2.12 in our case.
For any $n \in \mathbb{N}$, consider $QAP(T^{(n)}, B^{(n)})$, where $B^{(n)}$ is as described above. We have seen that for each $\pi \in \mathcal{S}_n$ the following equality holds: $N_\pi(k_n) = Z(T^{(n)}, B^{(n)}, \pi)$, where the $Z(T^{(n)}, B^{(n)}, \pi)$ is the objective function of $QAP(T^{(n)}, B^{(n)})$

Now, let us reformulate $QAP(T^{(n)}, B^{(n)})$ to fit in the general scheme of combinatorial optimization problems introduced in 1.3.1. The ground sets $\mathcal{E}_n$ are given as follows:

$$E_n = \{(i,j,k,l) \colon 1 \le i,j,k,l \le n\,,\, k_n \in P(k,l)\}\ ,$$

where $P(k,l)$ is the path from vertex v_k to v_l in $G^{(n)}$.

A feasible solution X_π corresponding to permutation $\pi \in \mathcal{S}_n$ is a subset of $\mathcal{E}_n$ of the form

$$X_\pi = \{(\pi(i), \pi(j), i, j) \colon k_n \in P(i,j)\}\ .$$

The set of the feasible solutions is then clearly given as $\mathcal{F}_n = \{X_\pi \colon \pi \in \mathcal{S}_n\}$. The cost function $f \colon \mathcal{E}_n \to \mathbb{R}^+$ maps each 4-tuple $(i,j,k,l) \in \mathcal{E}_n$ to the entry $t_{ij}^{(n)}$ of $T^{(n)}$.

With these definitions, the conditions of Theorem 2.12 are trivially fulfilled. In particular, we have $|\mathcal{F}_n| = n!$ and the cardinality of each feasible solutions equals the number of one-entries of matrix $B^{(n)}$. According to Lemma 7.16, these number is $\Omega(n^2)$. Thus, the condition

$$\lim_{n\to\infty} \frac{\ln |\mathcal{F}_n|}{|X_\pi|} = 0$$

is fulfilled. Applying Theorem 2.12 we get now that

$$\forall \epsilon > 0\,, \qquad \lim_{n\to\infty} P\left\{ \frac{Z\left(T^{(n)}, B^{(n)}, \phi_{war}^{(n)}\right)}{Z\left(T^{(n)}, B^{(n)}, \phi_{opt}^{(n)}\right)} \le 1 + \epsilon \right\} = 1\,,$$

where $\phi_{opt}^{(n)}$ is an optimal solution to $QAP(T^{(n)}, B^{(n)})$ and $\phi_{wor}^{(n)}$ is a worst solution to $QAP(T^{(n)}, B^{(n)})$. Considering the following two equalities completes the proof.

$$\max\{N_\pi(k_n) \colon \pi \in \mathcal{S}_n\} = Z(T^{(n)}, B^{(n)}, \phi_{war}^{(n)})$$

$$\min\{N_\pi(k_n) \colon \pi \in \mathcal{S}_n\} = Z(T^{(n)}, B^{(n)}, \phi_{opt}^{(n)})$$

■

Proof of Theorem 7.14. Consider the following equality:

$$\frac{CAP\left(T^{(n)}, G^{(n)}, \pi_{war}^{(n)}\right)}{CAP\left(T^{(n)}, G^{(n)}, \pi_{opt}^{(n)}\right)} = \frac{CAP\left(T^{(n)}, G^{(n)}, \pi_{war}^{(n)}\right)}{\max\{N_\pi(k_n) \colon \pi \in \mathcal{S}_n\}} \cdot \frac{\min\{N_\pi(k_n) \colon \pi \in \mathcal{S}_n\}}{CAP\left(T^{(n)}, G^{(n)}, \pi_{opt}^{(n)}\right)} \cdot \frac{\max\{N_\pi(k_n) \colon \pi \in \mathcal{S}_n\}}{\min\{N_\pi(k_n) \colon \pi \in \mathcal{S}_n\}} \tag{7.29}$$

Lemma 7.18 shows that the third term above approaches 1 with probability 1, as the size n of the problem tends to infinity. Lemma 7.17 implies that the same holds for the two other terms also and this completes the proof.

Indeed, the following inequalities hold:

$$\frac{CAP\left(T^{(n)}, B^{(n)}, \pi_{wor}^{(n)}\right)}{\max\{N_\pi(k_n) : \pi \in \mathcal{S}_n\}} \le \frac{CAP\left(T^{(n)}, B^{(n)}, \pi_{wor}^{(n)}\right)}{N_{\pi_{wor}^{(n)}}(k_n)}$$

$$\frac{\min\{N_\pi(k_n) : \pi \in \mathcal{S}_n\}}{CAP\left(T^{(n)}, B^{(n)}, \pi_{opt}^{(n)}\right)} \le \frac{N_{\pi_{opt}^{(n)}}(k_n)}{CAP\left(T^{(n)}, B^{(n)}, \pi_{opt}^{(n)}\right)}$$

Now, applying Lemma 7.17 for $\pi_{opt}^{(n)}$ and $\pi_{wor}^{(n)}$, for all $\epsilon > 0$ we get

$$\lim_{n\to\infty} P\left\{\frac{CAP\left(T^{(n)}, G^{(n)}, \pi_{opt}^{(n)}\right)}{N_{\pi_{opt}^{(n)}}(k_n)} \le 1+\epsilon\right\} = 1$$

$$\lim_{n\to\infty} P\left\{\frac{CAP\left(T^{(n)}, G^{(n)}, \pi_{wor}^{(n)}\right)}{N_{\pi_{wor}^{(n)}}(k_n)} \le 1+\epsilon\right\} = 1$$

The last four assertions imply that the two first ratios in (7.29) tend to 1 with probability equal to 1, as the size of the problem tends to infinity. ■

7.5 Conclusions and future research on CAPs

In this chapter we considered the communication assignment problem which arises along with the location of stations in a local area computer network. We distinguished polynomially solvable cases and NP-hard restricted versions of the problem. It turns out that the problem remains NP-hard also for simple underlying networks such as paths and cycles. For the case when the underlying network is a tree a branch and bound algorithm is proposed. Due to expensive lower bound computation this algorithm is unable to solve problems of size larger than 11 within acceptable time limits. The quality of the lower bounds, which is not very good in the early stages of the search, affects the robustness of the method. Namely, the performance of the algorithm strongly depends on the initial solution quality. Finally, the asymptotic behavior of CAPs on trees is investigated. It is proven that also this problem shows the same "strange" behavior as the QAP and the BiQAP, when the size of the problem increases. Namely, under natural probabilistic assumptions, the ratio between the best and the worst objective function values converges to one as the size of the problem approaches infinity. Preliminary computational results show that such a behavior can be detected also for relatively small problems (the corresponding trees have about 60 vertices).

There remain a lot of open questions related to the CAP. First, from a theoretical point of view the role of the structure of the communication matrix with respect to the complexity of the problem is not clear. More precisely, we investigated restricted versions of the problem where the restriction are put on the network structure. What about restrictions on the structure of the communication matrix? Do such restrictions lead to polynomially solvable cases of the problem? Can different properties of the communication matrix be exploited to improve the performance of exact algorithms? In this aspect, CAPs with sparse communication matrices would be of practical relevance. As a second open problem of obvious relevance let us mention here the generation of CAP test examples with known optimal value.

From a practical point of view, the focus is on heuristic approaches. We are working on testing the well known metaheuristics, such as simulated annealing and tabu search on CAPs. Another probably fruitful approach could be the design of problem specific heuristics, that is heuristics which really exploit the combinatorial properties of this problem. Let us notice however that our research in this direction is only in the very first steps.

Bibliography

[1] D. Adolphson and T. C. Hu, Optimal linear ordering, *SIAM Journal on Applied Mathematics* **25**, 1973, 403-423.

[2] R. K. Ahuja, J. B. Orlin and A. Tivari, A greedy genetic algorithm for the quadratic assignment problem, *Working paper*, Sloan School of Management, MIT, 1995.

[3] P. Ashar, S. Devadas and R. A. Newton, *Sequential Logic Synthesis*, Kluwer Academic Publishers, Boston, 1992.

[4] A. A. Assad and W. Xu, On lower bounds for a class of quadratic $0, 1$ programs, *Operations Research Letters* **4**, 1985, 175-180.

[5] E. H. L. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, John Wiley & Sons, 1989.

[6] A. S. Azarionok, Packing graphs of width two, *Vesti Akad. Navuk BSSR, Physics and Math. Sci. Minsk* **3**, 1987, 9-16. (in Russian)

[7] E. Balas and J. B. Mazzola, Quadratic $0-1$ programming by a new linearization, presented at *the Joint ORSA/TIMS National Meeting*, 1980, Washington D.C.

[8] F. Barahona and A. R. Mahjoub, On the cut polytope, *Mathematical Programming* **36**, 1986, 157-173.

[9] F. Barahona, J. Foulupt and A. R. Mahjoub, Composition of graphs and polyhedra iv: acyclic spanning subgraphs, *Mimeo*, 1990.

[10] R. Battiti and G. Tecchiolli, The reactive tabu search, *ORSA Journal on Computing* **6**, 1994, 126-140.

[11] R. Battiti and G. Tecchiolli, Simulated annealing and tabu search in the long run: A comparison on QAP tasks, *Computers and Mathematics with Applications* **28**, 1994, 1-8.

[12] M. S. Bazaraa and O. Kirca, Branch and bound based heuristics for solving the quadratic assignment problem, *Naval Research Logistics Quarterly* **30**, 1983, 287-304.

[13] M. S. Bazaraa and H. D. Sherali, Benders' partitioning scheme applied to a new formulation of the quadratic assignment problem, *Naval Research Logistics Quarterly* **27**, 1980, 29-41.

[14] M. S. Bazaraa and H. D. Sherali, On the use of exact and cutting plane methods for the quadratic assignment problem, *Journal of the Operations Research Society* **33**, 1982, 991-1003.

[15] J. F. Benders, Partitioning procedures for solving mixed variable programming problems, *Numerische Mathematik* **4**, 1962, 238-252.

[16] B. Boffey and J. Karkazis, Location of transfer centres of a communication network with proportional traffic, *Journal of the Operational Research Society* **40**, 1989, 729-734.

[17] S. H. Bokhari, On the mapping problem, *IEEE Transactions on Computers* **C-30**, 1981, 207-214.

[18] B. Bollobás, *Extremal Graph Theory*, Academic Press, 1978.

[19] B. Bollobás and S. E. Eldrige, Packing of graphs and applications to computational complexity, *Journal of Combinatorial Theory (B)* **25**, 1978, 105-124.

[20] P. A. Bruijs, On the quality of heuristic solutions to a 19×19 quadratic assignment problem, *European Journal of Operational Research* **17**, 1984, 21-30.

[21] R. E. Burkard, Die Störungsmethode zur Lösung quadratischer Zuordnungsprobleme, *Operations Research Verfahren* **16**, 1973, 84-108.

[22] R. E. Burkard, Locations with Spatial Interactions: The Quadratic Assignment Problem, in *Discrete Location Theory*, P.B. Mirchandani and R.L. Francis, eds., John Wiley & Sons, 1991, pp. 387-437.

[23] R. E. Burkard and T. Bönniger, A heuristic for quadratic Boolean programs with applications to quadratic assignment problems, *European Journal of Operational Research* **13**, 1983, 374-386.

[24] R. E. Burkard and E. Çela, Heuristics for biquadratic assignment problems and their computational comparison, *European Journal of Operational Research* **83**, 1995, 283-300.

[25] R. E. Burkard, E. Çela, V. M. Demidenko, N. N. Metelski and G. J. Woeginger, Easy and hard cases of the quadratic assignment problem: A survey, *Working paper*, Graz University of Technology, 1995.

[26] R. E. Burkard, E. Çela and B. Klinz, On the biquadratic assignment problem, in *Quadratic Assignment and Related Problems*, Proceedings of the DIMACS Workshop on Quadratic Assignment Problems, P.Pardalos and H.Wolkowicz, eds., *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **16**, 1994, 117-144.

[27] R. E. Burkard, E. Çela, G. Rote and G. J. Woeginger, The quadratic assignment problem with an anti-Monge and a Toeplitz matrix: Easy and hard cases, *SFB Report* No. **34**, Institute of Mathematics, Graz University of Technology, 1995.

[28] R. E. Burkard, E. Çela and G. J. Woeginger, A minimax assignment problem in treelike communication networks, to appear in *European Journal of Operational Research.*

[29] R. E. Burkard and U. Derigs, Assignment and Matching Problems: Solution Methods with FORTRAN- Programs, *Lecture Notes in Economics and Mathematical Systems* **184**, Springer, Berlin, 1980.

[30] R. E. Burkard and U. Fincke, Probabilistic asymptotic properties of some combinatorial optimization problems, *Discrete Applied Mathematics* **12**, 1985, 21–29.

[31] R. E. Burkard, S. E. Karisch and F. Rendl, QAPLIB - a quadratic assignment problem library, *European Journal of Operational Research* **55**, 1991, 115-119, updated version Feb. 1993.

[32] R. E. Burkard, B. Klinz and R. Rudolf, Perspectives of Monge properties in optimization, to appear in *Discrete Applied Mathematics.*

[33] R. E. Burkard and J. Offermann, Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme, *Zeitschrift für Operations Research* **21**, 1977, B121-B132.

[34] R. E. Burkard and F. Rendl, A thermodynamically motivated simulation procedure for combinatorial optimization problems, *European Journal of Operational Research* **17**, 1983, 169-174.

[35] V. N. Burkov, M. I. Rubinstein and V. B. Sokolov, Some problems in optimal allocation of large-volume memories, *Avtomatika i Telemekhanika* **9**, 1969, 83-91. (in Russian)

[36] P. Carraresi and F. Malucelli, A new lower bound for the quadratic assignment problem, *Operations Research* **40**, 1992, Suppl. No. **1**, S22-S27.

[37] E. Çela and G. J. Woeginger, A note on the maximum of a certain bilinear form, *SFB Report* No. **8**, Institute of Mathematics, Graz University of Technology, 1994.

[38] V. Černý, Thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm, *Journal of Optimization Theory and Applications* **45**, 1985, 41-51.

[39] J. Chakrapani and J. Skorin-Kapov, Massively parallel tabu search for the quadratic assignment problem, *Annals of Operations Research* **41**, 1993, 327-342.

[40] N. Christofides and E. Benavent, An exact algorithm for the quadratic assignment problem on a tree, *Operations Research* **37**, 1989, 760-768.

[41] N. Christofides and M. Gerrard, Special cases of the quadratic assignment problem, *Management Science Research Report* **391**, Carnegie-Mellon University, 1976.

[42] N. Christofides and M. Gerrard, A graph theoretic analysis of bounds for the quadratic assignment problem, in *Studies on Graphs and Discrete Programming*, P.Hansen, ed., North Holland, 1981, pp. 61-68.

[43] F. R. K. Chung, On optimal linear arrangements of trees, *Computers and Mathematics with Applications* **10**, 1984, 43-60.

[44] J. Clausen and M. Perregård, Solving large quadratic assignment problems in parallel, to appear in *Computational Optimization and Applications*.

[45] D. T. Connolly, An improved annealing scheme for the QAP, *European Journal of Operational Research* **46**, 1990, 93-100.

[46] K. Conrad, *Das quadratische Zuordnungsproblem und zwei seiner Spezialfälle*, Mohr-Siebeck, Tübingen, 1971.

[47] D. G. Corneil, Y. Perl and L. K. Stewart, A linear recognition algorithm for cographs, *SIAM Journal on Computing* **14**, 1985, 926-934.

[48] L. Davis, *Genetic Algorithms and Simulated Annealing*, Morgan Kaufmann Publishers, 1987.

[49] J. W. Dickey and J. W. Hopkins, Campus building arrangement using TOPAZ, *Transportation Research* **6**, 1972, 59-68.

[50] J. Edmonds and R. Giles, A min-max relation for submodular functions on graphs, *Annals of Discrete Mathematics* **1**, 1977, 185-204.

[51] C. S. Edwards, The derivation of a greedy approximator for the Koopmans-Beckmann quadratic assignment problem, in *Proceedings of the CP77 Combinatorial Programming Conference*, 1977, 55-86.

[52] C. S. Edwards, A branch and bound algorithm for the Koopmans-Beckmann quadratic assignment problem, *Mathematical Programming Study* **13**, 1980, 35-52.

[53] A. N. Elshafei, Hospital layout as a quadratic assignment problem, *Operations Research Quarterly* **28**, 1977, 167-179.

[54] B. Eschermann, State assignment methods for synchronous sequential circuits, in *Progress in Computer Aided VLSI Design*, G.Zobrist, ed., Ablex, Norwood, 1990.

[55] B. Eschermann and H.-J. Wunderlich, Optimized synthesis of self-testable finite state machines, *20th International Symposium on Fault-Tolerant Computing (FTCS 20)*, Newcastle upon Tyne, 1990.

[56] S. Evan and Y. Shiloach, NP-Completeness of several arrangement problems, *Technical Report No.* **43**, Computer Science Department, Technion, Haifa, Israel, 1975.

[57] U. Faigle and W. Kern, Note on the convergence of simulated annealing algorithms, *SIAM Journal on Control and Optimization* **29**, 1991, 153-159.

[58] T. Feo and M. G. C. Resende, Greedy randomized adaptive search procedures, *Journal of Global Optimization* **6**, 1995, 109-133.

[59] G. Finke, R. E. Burkard and F. Rendl, Quadratic assignment problems, *Annals of Discrete Mathematics* **31**, 1987, 61-82.

[60] C. Fleurent and J. Ferland, Genetic hybrids for the quadratic assignment problem, in *Quadratic Assignment and Related Problems*, Proceedings of the DIMACS Workshop on Quadratic Assignment Problems, P.Pardalos and H.Wolkowicz, eds., *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **16**, 1994, 173-187.

[61] A. Frank, How to make a digraph strongly connected, *Combinatorica* **1**, 1981, 145-153.

[62] J. B. G. Frenk, M. van Houweninge and A. H. G. Rinnooy Kan, Asymptotic properties of the quadratic assignment problem, *Mathematics of Operations Research* **10**, 1985, 100-116

[63] A. M. Frieze and J. Yadegar, On the quadratic assignment problem, *Discrete Applied Mathematics* **5**, 1983, 89-98.

[64] A. M. Frieze, J. Yadegar, S. El-Horbaty and D. Parkinson, Algorithms for assignment problems on an array processor, *Parallel Computing* **11**, 1989, 151-162.

[65] H. N. Gabow, A representation for crossing set families with applications to submodular flow problems, in *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1993.

[66] H. N. Gabow and R. E. Tarjan, A linear time algorithm for a special case of disjoint set union, in *Proceedings of the 15th Annual ACM Symposium on Theory of Computing*, 1983, 246-251.

[67] G. Gallo, M. D. Grigoriadis and R. E. Tarjan, A fast parametric maximum flow algorithm and applications, *SIAM Journal on Computing* **18**, 1989, 30-55.

[68] M. R. Garey and D. S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*, W.H. Freeman and Company, 1979.

[69] J. W. Gavett and N. V. Plyter, The optimal assignment of facilities to locations by branch and bound, *Operations Research* **14**, 1966, 210-232.

[70] F. Gavril, Some NP-complete problems on graphs, in *Proceedings of the 11th Conference on Information Sciences and Systems*, Johns Hopkins University, Baltimore, Maryland, 1977, 91-95.

[71] A. M. Geoffrion and G. W. Graves, Scheduling parallel production lines with changeover costs: Practical applications of a quadratic assignment/LP approach, *Operations Research* **24**, 1976, 595-610.

[72] P. C. Gilmore, Optimal and suboptimal algorithms for the quadratic assignment problem, *SIAM Journal on Applied Mathematics* **10**, 1962, 305-313.

[73] F. Glover, Tabu search - Part I, *ORSA Journal on Computing* **1**, 1989, 190-206.

[74] F. Glover, Tabu search - Part II, *ORSA Journal on Computing* **2**, 1989, 4-32.

[75] F. Glover, M. Laguna, E. Taillard, D. de Werra eds., Tabu search, *Annals of Operations Research* **41**, 1993.

[76] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[77] M. K. Goldberg and I. A. Klipker, Minimal placing of trees on a line, *Technical Report*, Physico-Technical Institute of Low Temperatures, Academy of Sciences of the Ukrainian SSR, USSR, 1976. (in Russian)

[78] R. E. Gomory and T. C. Hu, Multi-terminal network flows, *SIAM Journal on Applied Mathematics* **9**, 1961, 551-570.

[79] G. W. Graves and A. B. Whinston, An algorithm for the quadratic assignment problem, *Management Science* **17**, 1970, 453-471.

[80] M. Grötschel, Approaches to hard combinatorial optimization problems, in *Modern Applied Mathematics: Optimization and Operations Research*, B. Korte, ed., North Holland, 1982, 437-515.

[81] M. Grötschel, M. Jünger and G. Reinelt, On the acyclic subgraph polytope, *Mathematical Programming* **33**, 1985, 28-42.

[82] M. Hanan and J. M. Kurtzberg, A review of the placement and quadratic assignment problems, *SIAM Review* **14**, 1972, 324-342.

[83] S. W. Hadley, F. Rendl and H. Wolkowicz, Bounds for the quadratic assignment problem using continuous optimization techniques, in *Integer Programming and Combinatorial Optimization*, University of Waterloo Press, 1990, pp. 237-248.

[84] S. W. Hadley, F. Rendl and H. Wolkowicz, Symmetrization of nonsymmetric quadratic assignment problems and the Hoffman-Wielandt inequality, *Linear Algebra and its Applications* **167**, 1992, 53-64.

[85] S. W. Hadley, F. Rendl and H. Wolkowicz, A new lower bound via projection for the quadratic assignment problem, *Mathematics of Operations Research* **17**, 1993, 727-739.

[86] G. H. Hardy, J. E. Littlewood and G. Pólya, The maximum of a certain bilinear form, *Proceedings of the London Mathematical Society* **25**, 1926, 265–282.

[87] G. H. Hardy, J. E. Littlewood and G. Pólya, *Inequalities*, Cambridge University Press, Cambridge, 1952, pp. 260-270.

[88] S. M. Hedetniemi, S. T. Hedetniemi and P. J. Slater, A note on packing trees into K_N, *Ars Combinatorica* **11**, 1981, 149-153.

[89] D. R. Heffley, Assigning runners to a relay team, in *Optimal Strategies in Sports*, S.P. Ladany and R.E. Machol, eds., North-Holland, Amsterdam, 1977, pp. 169-171.

[90] C. H. Heider, A computationally simplified pair exchange algorithm for the quadratic assignment problem, Pap. No. **101**, *Center for Naval Analysis*, Arlington, Virginia, 1972.

[91] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, University of Michigan Press, 1975.

[92] J. E. Hopcroft and R. E. Tarjan, Efficient planarity testing, *Journal of ACM* **21**, 1974, 549-568.

[93] L. J. Hubert, *Assignment methods in combinatorial data analysis*, Marcel Dekker, Inc., New York, NY 10016, 1987.

[94] M. Jünger, *Polyhedral Combinatorics and the Acyclic Subdigraph Problem*, Heldermann Verlag, Berlin, 1985.

[95] D. S. Johnson, C. H. Papadimitriou and M. Yannakakis, How easy is local search, *Journal of Computer and System Sciences* **37**, 1988, 79-100.

[96] B. K. Kaku and G. L. Thompson, An exact algorithm for the general quadratic assignment problem, *European Journal of Operational Research* **2**, 1986, 382-390.

[97] S. E. Karisch, Nonlinear Approaches for Quadratic Assignment and Graph Partition Problems, *Ph.D. Thesis*, Graz University of Technology, 1995.

[98] S. E. Karisch, F. Rendl and H. Wolkowicz, Trust regions and relaxations for the quadratic assignment problem, in *Quadratic Assignment and Related Problems*, Proceedings of the DIMACS Workshop on Quadratic Assignment Problems, P.Pardalos and H.Wolkowicz, eds., *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **16**, 1994, 199-220.

[99] J. Karkazis, A minimax assignment problem on a linear communication network, *Belgian Journal of Operations Research, Statistics and Computer Science* **33**, 1993, 5-17.

[100] R. M. Karp, Reducibility among combinatorial problems, in *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York, 1972, 85-103.

[101] V. Karzanov, On the minimal number of arcs of a digraph meeting all its directed cutsets, (abstract), *Graph Theory Newsletters* **8**, 1979.

[102] L. Kaufman and F. Broeckx, An algorithm for the quadratic assignment problem using Benders' decomposition, *European Journal of Operational Research* **2**, 1978, 204-211

[103] B. Kernighan and S. Lin, An efficient heuristic procedure for partitioning graphs, *Bell Systems Journal* **49**, 1972, 291-307.

[104] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by simulated annealing, *Science* **220**, 1983, 671-680.

[105] T. C. Koopmans and M. J. Beckmann, Assignment problems and the location of economic activities, *Econometrica* **25**, 1957, 53-76.

[106] N. M. Korneenko, On the complexity of computing a distance between graphs, *Vesti of Belarus Acad. Sci.* **1**, 1981, 10-13. (in Russian)

[107] J. Krarup and P. M. Pruzan, Computer-aided layout design, *Mathematical Programming Study* **9**, 1978, 75-94.

[108] A. V. Krushevski, The linear programming problem on a permutation group, in *Proceedings of the seminar on methods of mathematical modelling and theory of electrical circuits*, Institut of Cybernetics of the Academy of Sciences of Ukraine, No. **3**, 1964, 364–371. (in Russian)

[109] A. V. Krushevski, Extremal problems for linear forms on permutations and applications, in *Proceedings of the seminar on methods of mathematical modelling and theory of electrical circuits*, Institut of Cybernetics of the Academy of Sciences of Ukraine, No. **3**, 1965, 262–269. (in Russian)

[110] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*, D.Reidel Publishing Company, 1988.

[111] A. M. Land, A problem of assignment with interrelated costs, *Operations Research Quarterly* **14**, 1963, 185-198.

[112] G. Laporte and H. Mercure, Balancing hydraulic turbine runners: a quadratic assignment problem, *European Journal of Operational Research* **35**, 1988, 378-382.

[113] P. S. Laursen, Simple approaches to parallel branch and bound, *Parallel Computing* **19**, 1993, 143-152.

[114] P. S. Laursen, Simulated annealing for the QAP - Optimal tradeoff between simulation time and solution quality, *European Journal of Operational Research* **69**, 1993, 238-243.

[115] E. L. Lawler, The quadratic assignment problem, *Management Science* **9**, 1963, 586-599.

[116] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, *The Traveling Salesman Problem*, John Wiley & Sons, Chichester, 1985.

[117] Y. Li and P. M. Pardalos, Generating quadratic assignment test problems with known optimal permutations, *Computational Optimization and Applications* **1**, 1992, 163-184.

[118] Y. Li, P. M. Pardalos, K. G. Ramakrishnan and M. G. C. Resende, Implementation of a variance reduction based lower bound in a branch and bound algorithm for the quadratic assignment problem, *to appear in SIAM Journal on Optimization.*

[119] Y. Li, P. M. Pardalos, K. G. Ramakrishnan and M. Resende, Lower bounds for the quadratic assignment problem, *Annals of Operations Research* **50**, 1994, 387-410.

[120] Y. Li, P. M. Pardalos and M. Resende, A greedy randomized adaptive search procedure for the quadratic assignment problem, in *Quadratic Assignment and Related Problems*, Proceedings of the DIMACS Workshop on Quadratic Assignment Problems, P.Pardalos and H.Wolkowicz, eds., *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **16**, 1994, 237-261.

[121] L. Lovász, On two minimax theorems in graph, *Journal of Combinatorial Theory (B)* **21**, 1976, 96-103.

[122] C. L. Lucchesi, A Minimax Inequality for Directed Graphs, *Ph.D. Thesis*, University of Waterloo, Waterloo, Ontario, 1976.

[123] C. L. Lucchesi and D. H. Younger, A minimax theorem for directed graphs, *Journal of the London Mathematical Society (2)* **17**, 1978, 369-374.

[124] F. Malucelli, Quadratic Assignment Problems: Solution Methods and Applications, *Ph.D. Thesis*, Universitá degli Studi di Pisa, Dipartimento di Informatica, Pisa, Italy, 1993.

[125] M. M. Mano, *Computer System Architecture*, 2nd edition, Prentice Hall, Englewood Cliffs, N.J., 1982.

[126] T. Mautor and C. Roucairol, A new exact algorithm for the solution of quadratic assignment problems, *Discrete Applied Mathematics* **55**, 1994, 281-293.

[127] N. N. Metelski, On extremal values of quadratic forms on symmetric groups, *Vesti Akad. Navuk BSSR, Ser. Fiz.-Mat. Navuk* **6**, 1972, 107–110. (in Russian)

[128] N. N. Metelski, Some special problems on placements and packings of graphs, Abstract in *Vesti Akad. Navuk BSSR, Physics and Math. Sci. Minsk* **5**, 1984, p. 110. (in Russian)

[129] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, Equations of state calculations by fast computing machines, *Journal of Chem. Physics* **21**, 1953, 1087-1092.

[130] P. B. Mirchandani and T. Obata, Algorithms for a class of quadratic assignment problems, presented at *the Joint ORSA/TIMS National Meeting*, 1979, New Orleans.

[131] G. De Micheli, R. K. Brayton and A. Sangiovanni-Vincentelli, Symbolic design of combinational and sequential logic circuits implemented by two-level logic macros, *IEEE Transactions on Computer Aided Design* **5**, 1986, 597-616.

[132] J. Mosevich, Balancing hydraulic turbine runners — a discrete combinatorial optimization problem, *European Journal of Operational Research* **26**, 1986, 202–204.

[133] K. A. Murthy and P. Pardalos, A polynomial time approximation algorithm for the quadratic assignment problem, *Technical Report* CS-**33**-**90**, 1990, Pennsylvania State University.

[134] K. A. Murthy, P. Pardalos and Y. Li, A local search algorithm for the quadratic assignment problem, *Informatica* **3**, 1992, 524-538.

[135] H. Müller-Merbach, *Optimale Reihenfolgen,* Springer Verlag, Berlin and New York, 1970, pp. 158-171.

[136] C. E. Nugent, T. E. Vollmann and J. Ruml, An experimental comparison of techniques for the assignment of facilities to locations, *Journal of Operations Research* **16**, 1969, 150-173.

[137] Z. Nutov and M. Penn, Minimum feedback arc set and maximum integral dicycle packing algorithms in $K_{3,3}$-free digraphs, *Technical Report*, Technion, Faculty of Industrial Engineering and Management, Haifa, Israel, June 1994.

[138] G. S. Palubeckis, Generation of quadratic assignment test problems with known optimal solutions, *U.S.S.R. Comput. Maths. Math. Phys.* **28**, 1988, 97-98. (in Russian)

[139] C. H. Papadimitriou and K. Steiglitz, Local Search, in *Combinatorial Optimization: Algorithms and complexity*, Prentice-Hall, Inc., Englewood Cliffs, NJ 07632, USA, 1982, pp. 454-481.

[140] C. H. Papadimitriou and D. Wolfe, The complexity of facets resolved, in *Proceedings of the Foundations of Computer Sciences*, 1985, 74-78.

[141] P. Pardalos and J. Crouse, A parallel algorithm for the quadratic assignment problem, in *Proceedings of the Supercomputing Conference 1989*, ACM Press, 1989, 351-360.

[142] P. Pardalos, F. Rendl and H. Wolkowicz, The quadratic assignment problem: A survey and recent developments, in *Quadratic Assignment and Related Problems*, Proceedings of the DIMACS Workshop on Quadratic Assignment Problems, P.Pardalos and H.Wolkowicz, eds., *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **16**, 1994, 1-42.

[143] U. Pferschy, R. Rudolf and G. J. Woeginger, Monge matrices make maximization manageable, *Operations Research Letters* **16**, 1994, 245–254.

[144] M. A. Pollatschek, N. Gershoni and Y. T. Radday, Optimization of the typewritter keyboard by computer simulation, *Angewandte Informatik* **10**, 1976, 438-439.

[145] V. R. Pratt, An $N \log N$ algorithm to distribute N records optimally in a sequential access file, in *Complexity of Computer Computations*, R.E.Miller and J.W.Thatcher, eds., Plenum Press, New York, 1972, 111-118.

[146] M. Queyranne, Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems, *Operations Research Letters* **4**, 1986, 231-234.

[147] V. Ramachandran, Finding a minimum feedback arc set in reducible flow graph, *Journal of Algorithms* **9**, 1988, 299-313.

[148] G. Reinelt, *The Linear Ordering Problem: Algorithms and Applications*, Heldermann Verlag, Berlin, 1985.

[149] F. Rendl, Ranking scalar products to improve bounds for the quadratic assignment problem, *European Journal of Operational Research* **20**, 1985, 363-372.

[150] F. Rendl, Quadratic assignment problems on series parallel digraphs, *Zeitschrift für Operations Research* **30**, 1986, 161–173.

[151] F. Rendl, Das Quadratische Zuordnungsproblem: Spezialfälle, Näherungsverfahren und Untere Schranken, *Habilitationschrift*, Institute of Mathematics, Graz University of Technology, 1988.

[152] F. Rendl and H. Wolkowicz, Applications of parametric programming and eigenvalue maximization to the quadratic assignment problem, *Mathematical Programming* **53**, 1992, 63-78.

[153] A. Rényi, *Probability Theory*, Amsterdam, North Holland, 1970.

[154] M. G. C. Resende, K. G. Ramakrishnan and Z. Drezner, Computing lower bounds for the quadratic assignment problem with an interior point algorithm for linear programming, to appear in *Operations Research* **43**, No. **5**, 1995.

[155] W. T. Rhee, A note on asymptotic properties of the quadratic assignment problem, *Operations Research Letters* **7**, 1988, 197-200.

[156] W. T. Rhee, Stochastic analysis of the quadratic assignment problem, *Mathematics of Operations Research* **16**, 1991, 223-239.

[157] C. Roucairol, A reduction method for quadratic assignment problems, *Operations Research Verfahren* **32**, 1979, 183-187.

[158] C. Roucairol, A parallel branch and bound algorithm for the quadratic assignment problem, *Discrete Applied Mathematics* **18**, 1987, 221-225.

[159] M. I. Rubinstein, Problems and methods in combinatorial programming, *Technical Report*, Institute of Control Sciences, Moscow, 1976. (in Russian)

[160] M. I. Rubinstein, *The Optimal Grouping of Related Objects*, NAUKA Publishers, Moscow, 1989. (in Russian)

[161] M. I. Rubinstein, manuscript, 1994.

[162] R. Rudolf and G. J. Woeginger, The cone of Monge matrices: Extremal rays and applications, *ZOR – Mathematical Methods of Operations Research* **42**, 1995, 161-168.

[163] S. Sahni and T. Gonzalez, P-complete approximation problems, *Journal of the Association of Computing Machinery* **23**, 1976, 555-565.

[164] N. Sauer and J. Spencer, Edge disjoint placement of graphs, *Journal of Combinatorial Theory (B)* **25**, 1978, 295-302.

[165] S. L. Savage, Some theoretical implications of local search, *Mathematical Programming* **10**, 1976, 354-366.

[166] A. Schäffer and M. Yannakakis, Simple local search problems that are hard to solve, *SIAM Journal on Computing* **20**, 56-87.

[167] D. Schlegel, Die Unwucht-optimale Verteilung von Turbinenschaufeln als quadratisches Zuordnungsproblem, *Ph.D. Thesis*, ETH Zürich, 1987.

[168] Y. Shiloach, A minimum linear arrangement algorithm for undirected trees, *SIAM Journal on Computing* **8**, 1979, 15-32.

[169] J. Skorin-Kapov, Tabu search applied to the quadratic assignment problem, *ORSA Journal on Computing* **2**, 1990, 33-45.

[170] W. Stallings, *Local Networks: An introduction*, MacMillan, New York, 1984.

[171] L. Steinberg, The backboard wiring problem: A placement algorithm, *SIAM Review* **3**, 1961, 37-50.

[172] F. Supnick, Extreme Hamiltonian lines, *Annals of Mathematics* **66**, 1957, 179-201.

[173] W. Szpankowski, Combinatorial optimization problems for which almost every algorithm is asymptotically optimal!, *Optimization* **33**, 1995, 359-367.

[174] E. Taillard, Robust taboo search for the quadratic assignment problem, *Parallel Computing* **17**, 1991, 443-455.

[175] E. Taillard, Comparison of iterative searches for the quadratic assignment problem, Centre de recherche sur les transports, Université de Montréal, Publication CRT-**989**, 1994, to appear in *Location science*.

[176] R. E. Tarjan, Testing flow graph reducibility, *Journal of Computer System Sciences* **9**, 1974, 355-365.

[177] D. E. Tate and A. E. Smith, A genetic approach to the quadratic assignment problem, to appear in *Computers and Operations Research*.

[178] B. B. Timofeev and V. A. Litvinov, On the extremal value of a quadratic form, *Kibernetica* **4**, 1969, 56-61. (in Russian)

[179] I. Ugi, J. Bauer, J. Friedrich, J. Gasteiger, C. Jochum and W. Schubert, Neue Anwendungsgebiete für Computer in der Chemie, *Angewandte Chemie* **91**, 1979, 99-111.

[180] J. Valdes, R. E. Tarjan and E. L. Lawler, The recognition of series parallel digraphs, *SIAM Journal on Computing* **11**, 1982, 471–506.

[181] D. Vanderbilt and S. G. Louie, A Monte Carlo simulated annealing approach to optimization over continuous variables, *Journal of Computational Physics* **56**, 1984, 259-271.

[182] J. A. A. van der Veen, Solvable Cases of the Travelling Salesman Problem with Various Objective Functions, *Ph.D. Thesis*, University of Groningen, 1992.

[183] D. H. West, Algorithm 608: Approximate solution of the quadratic assignment problem, *ACM Transcations on Mathematical Software* **9**, 1983, 461-466.

[184] M. R. Wilhelm and T. L. Ward, Solving quadratic assignment problems by "simulated annealing", *IIE Transcations*, 1987, 107-119.

[185] Q. F. Yang, R. E. Burkard, E. Çela and G. J. Wöginger, Hamiltonian cycles in circulant digraphs with two stripes, *SFB Report* No. **20**, Institute of Mathematics, Graz University of Technology, 1995.