Ante ĆUSTIĆ

Efficiently solvable special cases of multidimensional assignment problems

PHD THESIS

written to obtain the academic degree of a Doctor of Engineering Sciences

Doctoral studies of Engineering
at the doctoral school "Mathematics and Scientific Computing"

Graz University of Technology

Supervisor:
Ao.Univ.-Prof. Dipl.-Ing. Dr.techn. Bettina KLINZ
Institute of Optimization and Discrete Mathematics
(Math B)

Graz, July 2014

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

....	....
(date)	(signature)

Contents

List of Figures

Acknowledgements

I would like to thank my advisor Bettina Klinz for her patient guidance and support throughout my doctoral studies.

Special thanks goes to Gerhard Woeginger for sharing his time and ideas with me. I'm very grateful that I have had the opportunity to work with him. I'm further thankful to him for his kindness and hospitality during my stay at TU Eindhoven.

My special thanks are extended to Vladimir Deineko and Frits Spieksma for hosting me at University of Warwick and KU Leuven, and for introducing me to mathematicians that share my research interests.

I thank all the members of the DK Discrete Mathematics for making my time in Graz a joyful and valuable experience. I particularly thank my mentors Rainer Burkard and Wolfgang Woess for their help and support throughout the completion of this thesis.

Special thanks go to Franz Rendl and Reinhardt Euler for the time they invested into refereeing this thesis.

I further thank many people who have influenced me on my path to a doctoral degree in mathematics, in particular Slobodan Maroja, Janja Martinović and Vedran Krčadinac. Finally, I thank my family and friends for their constant support and care.

Chapter 1

Introduction

The *combinatorial optimization problems* considered in this thesis fit into the following framework. Given a ground set $E = \{1, \dots, n\}$, a set of *feasible solutions* $\mathcal{F} \subset 2^E$, and a real *objective value* function $f\colon 2^E \to \mathbb{R}$, find a feasible solution that minimizes f, i.e.

$$\text{Minimize } f(S), \quad \text{subject to } S \in \mathcal{F}.$$

Feasible solutions for which the minimal objective value is achieved are called *optimal solutions.*

Usually there is a *cost* $c(e)$ assigned to every ground element $e \in E$ and the objective value function is given by

$$f(S) := \sum_{e \in S} c(e). \tag{1.1}$$

Some combinatorial optimization problems can be solved by a polynomial time algorithm, i.e. they belong to the complexity class P. They are regarded as “efficiently solvable” and “easy” problems. Some problems that fall into this class are the *minimum spanning tree problem*, the *shortest path problem* and the *linear assignment problem.* Conversely, NP-hard problems are believed not to be polynomially solvable. Namely, the existence of a polynomial time algorithm for some NP-hard problem would imply P=NP, which is believed not to be true, see [41]. Some NP-hard combinatorial optimization problems are the *traveling salesman problem*, the *maximum independent set problem* and the *quadratic assignment problem.*

There are several approaches to handling NP-hard combinatorial optimization problems. For example, exponential time exact approaches like *branch and bound*, *cutting plane method* or *dynamic programing* can be applied. Furthermore, heuristics like *greedy*, *improvement heuristics*, *meta-heuristics* and *LP-relaxation* approaches can be developed to solve such problems in a suboptimal way. If a heuristic runs in polynomial time and outputs a solution with the objective value at most $\alpha > 1$ times the objective value of an optimal solution (at least $0 < \alpha < 1$ times for maximization problems), then such a heuristic is called *α-approximation algorithm.* Lastly, special cases of NP-hard problems can be investigated. Special cases investigation of the multidimensional assignment problems, with special emphasis on the three-dimensional assignment problems, is the central topic of this thesis.

If the considered combinatorial optimization problem is NP-hard, we can ask: Which subclasses of the problem allow fast solution (i.e. polynomial time algorithm) and which don't? Note that we do not ask for just one problem instance, but for an (infinite) class of problem instances which allow a fast algorithm. The knowledge about efficiently solvable special cases offers a better understanding of the underlying hard problem. By investigating the borderline between efficiently solvable and hard special cases we detect what makes the problem hard. We can use solvable special cases in heuristics or in approximating general problem instances by those with special properties. Furthermore, some problems in practice show a regular pattern which can be used for simplifying the solution process.

Some ways to arrive at special cases result from considering

- special cost structures,
- special geometry of the problem,
- special topology of the underlying graph structure,
- a special algorithm.

An important question in the field of special case investigation is the question of the fast recognition of special structures.

There is a relatively rich literature on a variety of special cases for many classes of NP-hard combinatorial optimization problems. For example, see

[43] and [20] for the surveys on the special cases of the traveling salesman problem, and [76] for the survey on the special cases for multidimensional assignment problems.

The rest of the thesis is organized as follows. In Chapter 2 we give a short overview of the assignment problems investigated in this thesis.

In Chapter 3 we investigate special cases of the axial 3-dimensional assignment problem (A3AP) that emerge from the geometric setting of the problem. We show that the clustering of three color points into three color triangles, where the objective value of the clustering is the sum of triangle perimeters, is NP-hard for a large class of distance metrics that, among others, generalize L_p metrics. Conversely, if we want to maximize the total perimeter instead of minimizing it, the problem becomes polynomially solvable for any fixed polyhedral metric. Furthermore, as a byproduct of our results we get that the analogous results hold for the bottleneck and one color versions of the problem. These results can be found in Ćustić et al. [29].

Chapter 4 is dedicated to special case investigation of the planar 3-dimensional assignment problem (P3AP). Unlike in the case of the A3AP, there are no nontrivial P3AP special case results in the literature. This thesis contributes to filling this void. The majority of our investigations deal with cost structures associated with Monge-like properties. We detect few NP-hard special cases and recognize some structural properties of the optimal solution that enable us to design polynomial dynamic programming algorithms. Most important results of Chapter 4 can be found in Ćustić et al. [30].

In Chapter 5 we aim to characterize all instances for which every feasible solution has the same objective value for a large class of multidimensional assignment problems and some other combinatorial optimization problems. Such instances define a trivially solvable special case. Furthermore, in the case of combinatorial optimization problems with the sum objective value function (1.1), such instances are in one-to-one correspondence with the so called *admissible transformations* that can be used to solve the problem or to find a lower bound, see Section 5.1. The main results of this chapter show that in the case of axial and planar d-dimensional assignment problems the constant objective value property can be characterized by sum-decomposable arrays. We prove that this is not the case for general multidimensional assignment problems by giving a counterexample. Other problems considered are the *transportation problem*, the *minimal spanning tree problem*, the *shortest path problem* and the *minimum weight maximum cardinality matching problem*. Results from this chapter can be found in Ćustić and Klinz [28].

Chapter 2

Preliminaries

Assignment problems are a class of combinatorial optimization problems where a family of objects arranged in disjoint sets need to be assigned to each other by some assignment rule. Every such assignment has the corresponding cost. The problem is to find the one with the smallest cost. For a comprehensive overview of the assignment problems see the book by Burkard et al. [21].

In this chapter we give a short overview of the assignment problems investigated in the latter chapters. In Section 2.1 the linear assignment problem is addressed. In Section 2.2 and Section 2.3 two three-dimensional assignment problem variants are presented. Finally, a generalization of these notions in higher dimensions is presented in Section 2.4.

2.1 Linear assignment problem

The *linear assignment problem* (LAP) is one of the most famous problems in linear programming and combinatorial optimization. Consider n tasks and n machines. Assume that for every task-machine pair there is a cost associated to it. The linear assignment problem models the problem of assigning each task to a different machine so that the corresponding cost objective is optimized. The cost objective can be formulated in various ways. The classical one is as the sum of all task-machine pair costs in the assignment. This yields the *linear sum assignment problem* (LSAP). Another one is as the maximum of all task-machine pair costs. This variant is called the *linear bottleneck assignment problem* (LBAP). In both LSAP and LBAP one wants to find the assignment with the minimal corresponding objective value. If the costs are the energy required by a machine to perform a task, then the LSAP models the problem of finding the assignment that minimizes the total energy consumption. If the costs are the time needed for a machine to perform a task, then the LBAP models the problem of finding the assignment that minimizes the time required for all tasks to be completed.

If we express the costs in a form of a matrix $C = (c_{ij})$ where c_{ij} is the cost of assigning task i to machine j, then the LAP is the problem of choosing n elements of C with the minimal objective value, such that exactly one element from each row and each column is chosen.

2	4	5	3
2	8	5	**2**
1	**4**	7	3
4	6	**5**	6

$\longleftrightarrow$

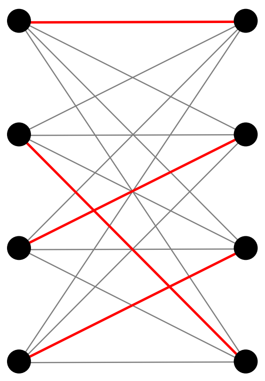

Figure 2.1: Matrix and graph theoretic representation of the LAP

The LAP can also be expressed in the graph theoretic language. Let $G = (U, V; U \times V)$ be a complete bipartite graph with $|U| = |V| = n$, where sets of vertices U and V correspond to the set of tasks and machines, respectively. Let the cost of performing a task i on a machine j be associated to the edge (i, j). Then the LAP is the problem of finding a perfect matching of a weighted graph G with the minimal objective value.

Every assignment is a bijective mapping of the n sized set of tasks to the n sized set of machines, that is, a permutation of the set $\{1, \ldots, n\}$. Therefore, there are $n!$ feasible solutions of the LAP. Let S_n denote a set of permutations of the set $\{1, \ldots, n\}$. Given a cost matrix $C = (c_{ij})$, the LSAP is the problem of finding a permutation $\varphi \in S_n$ that minimizes $\sum_{i=1}^{n} c_{i\varphi(i)}$, while the LBAP is the problem where the objective function

$\max_{i=1,\dots,n} c_{i\varphi(i)}$ is minimized.

2.1.1 Algorithms for the LAP

An integer linear programming formulation of the LSAP is as follows:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{2.1}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1, 2, \dots, n, \tag{2.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1, 2, \dots, n, \tag{2.3}$$

$$x_{ij} \in \{0, 1\} \qquad i, j = 1, 2, \dots, n. \tag{2.4}$$

In the case of LBAP we have

$$\min \max_{i,j=1,\dots,n} c_{ij} x_{ij} \tag{2.5}$$

instead of (2.1).

An $n \times n$ matrix $X = (x_{ij})$ whose entries fulfill (2.2) and (2.3) is called a *doubly stochastic* matrix. Doubly stochastic matrix with entries 0 and 1 is called a *permutation matrix*. Hence, every feasible solution of the LAP corresponds to a permutation matrix. The set of all doubly stochastic matrices forms the so-called *assignment polytope*. Birkhoff [16] showed that permutation matrices uniquely correspond to the vertices of the assignment polytope. Thus every doubly stochastic matrix can be written as a convex combination of permutation matrices. Due to this property, the LSAP can be solved using linear programming techniques.

The first combinatorial polynomial algorithm for the LSAP is the famous Hungarian method developed by Kuhn [55]. It is a primal-dual algorithm based on the results of Dénes Kőnig and Jenő Egerváry. The time complexity of the original version is $O(n^4)$, while today $O(n^3)$ implementations are known. Recently it was discovered that the Hungarian method was known to Jacobi [49] hundred years before it was discovered by Kuhn.

Many more algorithms for the LSAP were suggested, among which are the primal algorithm by Balinski and Gomory [11], the dual algorithm by Dinic and Kronrod [33], the *auction algorithm* by Bertsekas [15], simplex-based algorithms [3], pseudoflow algorithms [45], the decomposition algorithm by Kao et al. [51], etc. The best strongly polynomial time bound for the LSAP is still $O(n^3)$.

Next we consider algorithms for the LBAP. One natural approach to solving the LBAP is the *threshold algorithm.* It alternates between two phases. In the first phase a cost element c^* (the threshold value) is chosen and a *threshold matrix* $\bar{C} = (\bar{c}_{ij})$ is defined as

$$\bar{c}_{ij} = \begin{cases} 1 & \text{if } c_{ij} > c^*, \\ 0 & \text{otherwise.} \end{cases}$$

In the second phase it is verified if for the cost matrix $\bar{C}$ there exists an assignment with total cost 0. In other words, one must check whether the bipartite graph with edges that correspond to values 0 in the threshold matrix, contains a perfect matching. Then the smallest value c^* for which the corresponding bipartite graph contains a perfect matching is the optimum value of the LBAP. A good implementation of the threshold algorithm yields complexity of $O(n^{2.5}/\sqrt{\log n})$, which is a theoretically best bound for a dense LBAP (the number of nonzero coefficients c_{ij} is $O(n^2)$).

An algorithm for the LBAP that mimics the Hungarian method can be implemented using augmenting path methods, see [32].

2.2 Axial 3-dimensional assignment problem

Two standard ways to generalize the LAP to three dimensions are the *axial 3-dimensional assignment problem* which is addressed in this section, and the *planar 3-dimensional assignment problem* which will be addressed in Section 2.3. These names were first used by Schell in 1955, see [74].

2.2.1 Problem statement and applications

Given an $n \times n \times n$ array $C = (c_{ijk})$, the axial 3-dimensional assignment problem, A3AP for short, can be stated as follows. We ask for two permutations $\varphi, \psi \in S_n$ such that $\sum_{i=1}^{n} c_{i\varphi(i)\psi(i)}$ is minimal, i.e. the problem is:

$$\min_{\varphi,\psi \in S_n} \sum_{i=1}^{n} c_{i\varphi(i)\psi(i)}.$$

Every pair of permutations φ and ψ yields a feasible solution, hence there are $(n!)^2$ feasible solutions of the A3AP. The A3AP is also known in the literature as the axial 3-index assignment problem.

Integer linear programming formulation of the A3AP is as follows.

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} c_{ijk}x_{ijk} \tag{2.6}$$

$$\text{s.t.} \quad \sum_{i=1}^{n}\sum_{k=1}^{n} x_{ijk} = 1 \qquad i = 1, 2, \ldots, n \tag{2.7}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{n} x_{ijk} = 1 \qquad j = 1, 2, \ldots, n \tag{2.8}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ijk} = 1 \qquad k = 1, 2, \ldots, n \tag{2.9}$$

$$x_{ijk} \in \{0, 1\} \qquad i, j, k = 1, 2, \ldots, n \tag{2.10}$$

Every feasible solution of the A3AP can be seen as a set of n elements of C that contains exactly one element from every matrix obtained by fixing an index of C. This is depicted in Figure 2.2.

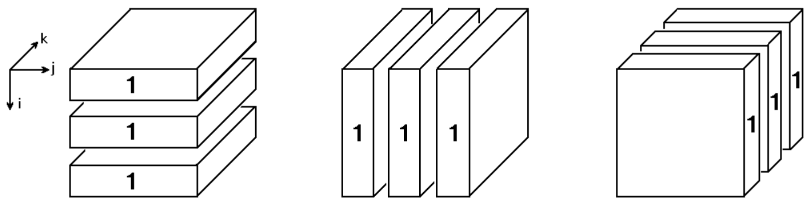

Figure 2.2: Depiction of the constraints (2.7), (2.8) and (2.9), respectively.

The *bottleneck axial 3-dimensional assignment problem*, the bottleneck-A3AP for short, is a version of the problem where the maximal element is minimized, i.e.

$$\min_{\varphi,\psi \in S_n} \max_{i=1,\ldots,n} c_{i\varphi(i)\psi(i)}. \tag{2.11}$$

The maximization version of the A3AP (max-A3AP) is also considered in the literature, for example in [23].

The (bottleneck) A3AP models the following problem. Given a set of n jobs, n workers and n machines such that c_{ijk} measures the cost of assigning a job i to a worker j on a machine k, we want to assign all jobs to different workers and machines so that the total (maximal) cost is minimized. The (bottleneck) A3AP has arisen in many applications some of which are: capital investment, dynamic facility location, satellite launching [63, 64], assembly of printed circuit boards [26], perishable production planning [7], scheduling a rolling mill [71], etc.

2.2.2 Complexity and algorithms

In contrast to the LAP, the A3AP is NP-hard as it is a straightforward generalization of the famous 3-Dimensional Matching which is shown to be NP-complete by Karp [52].

> 3-Dimensional Matching (3DM)
> Input: Three disjoint sets X, Y and Z ($n = |X| = |Y| = |Z|$) and a set of triples $T \subseteq X \times Y \times Z$.
> Question: Does there exists a subset of n triples $M \subseteq T$ such that every element of $X \cup Y \cup Z$ occurs in exactly one triple of M?

Given an instance of the 3DM one can define an $n \times n \times n$ array $C = (c_{ijk})$ as follows. Let elements of sets X, Y, Z correspond to indices i, j, k, respectively. Let c_{ijk} be equal to 0 if the triple that corresponds to (i, j, k) is in M, and 1 otherwise. Then, the given 3DM instance is a YES instance if and only if the optimal solution of the (bottleneck) A3AP with cost array C is 0. For the max-A3AP swap 0's and 1's in C. Hence, the (bottleneck) A3AP is NP-hard even when restricted to 0-1 cost arrays. Moreover, it is straightforward to see that for the minimization and the bottleneck-A3AP no polynomial algorithm can ever achieve a constant performance ratio unless P=NP. On the other hand, $\frac{1}{2}$-approximation algorithm for the max-A3AP is given by Arkin and Hassin [8].

The first branch and bound methods for the A3AP are due to Vlach [79] and Pierskalla [64]. A primal-dual implicit enumeration method based on a graph theoretic approach was designed by Hansen and Kaufman [47]. Polyhedral approaches were applied by Balas and Saltzman [10].

2.2.3 Special cases

Since the A3AP is NP-hard, its computational complexity when when restricted to special cases is of interest.

Many hard combinatorial optimization problems become efficiently solvable for Monge-like structures. Examples include the traveling salesman problem [43], the economic lot-sizing problems [2] and the axial d-dimensional assignment and transportation problems [22].

Figure 2.3: G. Monge

Basic Monge structures are *Monge matrices* which arise in many areas of mathematics [22] and were first investigated by the French mathematician Gaspard Monge.

Definition 2.1. An $n_1 \times n_2$ matrix $M = (m_{ij})$ is called Monge matrix if

$$m_{ij} + m_{kl} \leq m_{il} + m_{kj} \tag{2.12}$$

holds for all $1 \leq i < k \leq n_1,\ 1 \leq j < l \leq n_2$.

The notion of the Monge matrix can be generalized to $d > 2$ dimensions as follows.

Definition 2.2. An $n_1 \times n_2 \times \cdots \times n_d$ d-dimensional array $C = (c_{i_1 i_2 \cdots i_d})$ is called a *Monge array* if for $i_k = 1, \ldots, n_k$ and $j_k = 1, \ldots, n_k$, $k = 1, \ldots, d$, we have

$$c_{s_1 s_2 \cdots s_d} + c_{t_1 t_2 \cdots t_d} \leq c_{i_1 i_2 \cdots i_d} + c_{j_1 j_2 \cdots j_d}, \tag{2.13}$$

where $s_k = \min\{i_k, j_k\}$ and $t_k = \max\{i_k, j_k\}$, $j = 1, \ldots, d$.

Bottleneck Monge matrices and arrays are defined by replacing Monge properties (2.12) and (2.13) by

$$\max\{m_{ij}, m_{kl}\} \leq \max\{m_{il}, m_{kj}\} \tag{2.14}$$

and

$$\max\{c_{s_1 s_2 \cdots s_d}, c_{t_1 t_2 \cdots t_d}\} \leq \max\{c_{i_1 i_2 \cdots i_d}, c_{j_1 j_2 \cdots j_d}\}, \tag{2.15}$$

respectively.

For both the LAP and the A3AP an optimal solution is obtained by identity permutations in the case of cost structures being Monge matrices and Monge arrays, respectively, i.e. the following proposition holds [22].

Proposition 2.3. *Let ϵ_n denote the identity permutation on $\{1, \dots, n\}$ with $\epsilon_n(i) = i$.*

(i) ϵ_n is an optimal solution for the LSAP and the LBAP on $n \times n$ Monge matrices and bottleneck Monge matrices, respectively.

(ii) $\varphi = \epsilon_n$ and $\psi = \epsilon_n$ is an optimal solution for the (bottleneck) A3AP on $n \times n \times n$ (bottleneck) Monge arrays.

In Chapter 4 the planar 3-dimensional assignment problem on Monge-like structures is investigated.

Klinz and Woeginger [53] showed that identity permutations $\varphi = \epsilon_n$ and $\psi = \epsilon_n$ are also optimal for the bottleneck-A3AP if the cost array $C = (c_{ijk})$ satisfies the following *wedge condition*:

$$c_{lll} < \min\{c_{ijk} \colon 1 \le i \le l;\ i \le j, k \le n;\ j + k \ne 2i\}$$

for all $l = 1, \dots, n$.

The A3AP on decomposable cost arrays $C = (c_{ijk})$ of the form $c_{ijk} = a_i b_j d_k$, for some nonnegative reals a_i, b_i and d_i, is investigated in [42] and [23]. Since the cost array $\bar{c}_{ijk} = -a_i b_j d_k$ is a Monge array provided a_i, b_i and d_i are sorted in a nondecreasing order, the max-A3AP on C can be solved in polynomial time. In contrast, in [23] it is shown that the A3AP remains NP-hard.

Crama and Spieksma [27], Spieksma and Woeginger [77] and Polyakonskiy et al. [66] consider special cases arising from geometrical formulation of the A3AP. Given sets R, B and G of n *red*, *blue* and *green* points, respectively, let c_{ijk} denote the cost of the three-color triangle determined by points r_i, b_j and g_k. The problem of partitioning $R \cup B \cup G$ into n three-color triangles with the minimal total cost is exactly the A3AP. In [27], the sum of all three edges and the sum of the two longer edges are used as costs for triangles. Even if the point distances fulfill the triangle inequality the problem remains NP-hard. The authors designed $\frac{3}{2}$ and $\frac{4}{3}$-approximation algorithms for these two problem variants, respectively.

In [66] the authors show that both the minimization and maximization variant when the cost of a triangle is its perimeter is polynomially solvable, if the underlying metric space satisfies the so-called Kalmanson conditions; their results cover convex Euclidean point sets and tree metric spaces as a special case.

In [77], the points are placed on the plane, the cost of the triangle is its perimeter while the length of an edge is the Euclidean distance of the corresponding points. The problem remains NP-hard and it is so even if the sum of the areas of triangles instead of their perimeter is minimized.

Chapter 3 is dedicated to the investigation of the minimal and maximal total perimeter problem complexity when the points are placed in $\mathbb{R}^d$ for a large class of distance metrics.

2.3 Planar 3-dimensional assignment problem

The second classical way to define an assignment problem in the three-dimensional setting, along with the A3AP, is the planar 3-dimensional assignment problem, P3AP for short. The name planar 3-index assignment problem is also used in the literature.

2.3.1 Problem statement and applications

In the P3AP the goal is for a given $n \times n \times n$ cost array $C = (c_{ijk})$ to find n pairwise disjoint permutations $\varphi_1, \varphi_2, \dots, \varphi_n \in S_n$ so as to minimize the cost function

$$\sum_{k=1}^{n} \sum_{i=1}^{n} c_{i\varphi_k(i)k} \tag{2.16}$$

where two permutations $\varphi, \psi \in S_n$ are said to be disjoint if $\varphi(i) \neq \psi(i)$ for all $i \in \{1, \dots, n\}$.

An integer linear programming formulation of the P3AP is given below.

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijk} x_{ijk} \tag{2.17}$$

$$\text{s.t.} \quad \sum_{k=1}^{n} x_{ijk} = 1 \qquad i, j = 1, 2, \dots, n \tag{2.18}$$

$$\sum_{i=1}^{n} x_{ijk} = 1 \qquad j,k = 1,2,\dots,n \tag{2.19}$$

$$\sum_{j=1}^{n} x_{ijk} = 1 \qquad i,k = 1,2,\dots,n \tag{2.20}$$

$$x_{ijk} \in \{0,1\} \qquad i,j,k = 1,2,\dots,n \tag{2.21}$$

Every feasible solution of the P3AP can be seen as a set of n^2 elements of C that contains exactly one element from every *line* obtained by fixing two indices of C. This is depicted in Figure 2.4.

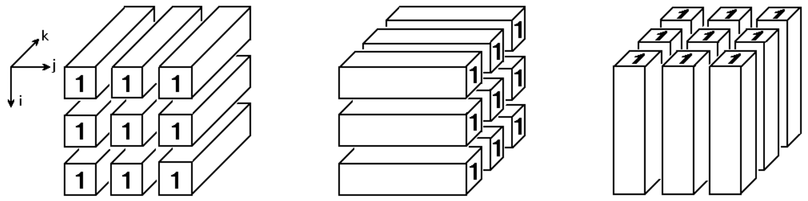

Figure 2.4: Depiction of the constraints (2.18), (2.20) and (2.19), respectively.

Furthermore, every feasible solution of the P3AP corresponds to and can be represented by a Latin square. Latin square of order n is an $n \times n$ table filled with integers from 1 to n such that every row and column contains every integer from 1 to n. To see this, let x be a feasible solution of the P3AP. By placing integer k into a cell in row i and column j if and only if $x_{ijk} = 1$ one obtains a Latin square. Namely, constraints (2.18) ensure that every cell contains exactly one integer, and constraints (2.20) and (2.19) ensure that every row and column contains all integers from 1 to n respectively. A Latin square representation of a feasible solution is depicted in Figure 2.5. Therefore, the number of feasible solutions of the P3AP with

$$\begin{aligned} x_{113} = x_{121} = x_{132} = x_{144} = 1 \\ x_{211} = x_{224} = x_{233} = x_{242} = 1 \\ x_{314} = x_{322} = x_{331} = x_{343} = 1 \\ x_{412} = x_{423} = x_{434} = x_{441} = 1 \end{aligned} \quad \longleftrightarrow$$

3	1	2	4
1	4	3	2
4	2	1	3
2	3	4	1

Figure 2.5: Latin square representation of a feasible solution.

an $n \times n \times n$ cost array is equal to the number of Latin squares of order

n. This number grows very quickly and no easily computable formula that computes it is known. Already for order 12 the number of Latin squares is unknown.

The *bottleneck planar 3-dimensional assignment problem* (bottleneck-P3AP) is a version of the problem where the maximal element is minimized.

The *p-layer planar 3-dimensional assignment problem*, p-P3AP for short, is the following variant / generalization of the P3AP. Given an $n \times n \times p$ cost array C where $p \in \{2, \dots, n\}$, the goal is to find p pairwise disjoint permutations $\varphi_1, \dots, \varphi_p$ so as to minimize the cost function

$$\sum_{k=1}^{p} \sum_{i=1}^{n} c_{i\varphi_k(i)k}.$$

Note that for $p = n$ the P3AP results. The associated integer linear program differs from the one for the P3AP by replacing the equality constraints in (2.18) by $\sum_{k=1}^{p} x_{ijk} \leq 1$ and setting the upper limit for the index k in the objective value (2.17) and in the constraints (2.19), (2.20) and (2.21) to p.

Feasible solutions of the p-P3AP can be represented as *Latin rectangles* or as *partial Latin squares*. A Latin rectangle L is an $m \times n$ table ($m \leq n$) filled with integers from 1 to n such that every row and every column contains every integer at most once. Note that an $n \times n$ Latin rectangle is a Latin square. A partial Latin square occurs if entries of the table can stay unfilled.

Observation 2.4. The set of feasible solutions of the p-P3AP is in one-to-one correspondence with

(i) the set of $p \times n$ Latin rectangles

(ii) the set of partial $n \times n$ Latin squares in which each of the integers $1, 2, \dots, p$ appears exactly n times and the integers $p + 1, \dots, n$ do not appear at all.

To see this, let x be a feasible solution of the p-P3AP. To show that (i) holds place integer i in column j and row k if and only if $x_{ijk} = 1$. To see that (ii) holds place integer k in row i and column j if and only if $x_{ijk} = 1$. See Figure 2.6 for an illustrating example with $n = 4$ and $p = 3$. The

$$\begin{aligned} x_{113} = x_{121} = x_{132} = 1 \\ x_{211} = x_{233} = x_{242} = 1 \\ x_{322} = x_{331} = x_{343} = 1 \\ x_{412} = x_{423} = x_{441} = 1 \end{aligned} \quad \longleftrightarrow \quad \begin{array}{|c|c|c|c|} \hline 2 & 1 & 3 & 4 \\ \hline 4 & 3 & 1 & 2 \\ \hline 1 & 4 & 2 & 3 \\ \hline \end{array} \quad \longleftrightarrow \quad \begin{array}{|c|c|c|c|} \hline 3 & 1 & 2 & \\ \hline 1 & & 3 & 2 \\ \hline & 2 & 1 & 3 \\ \hline 2 & 3 & & 1 \\ \hline \end{array}$$

Figure 2.6: Example for Latin rectangle/partial Latin square representation

partial Latin square representation uses $n \times n$ table regardless of the value of p, hence for small values of p the Latin rectangle representation is more compact. Note that when $p = n$ both the Latin rectangle and the partial Latin square representation results with a Latin square, but such two Latin squares will not coincide.

The (bottleneck) p-P3AP models the following timetabling problem: p groups of students need to be taught by n professors within n time slots. If c_{ijk} is the measure of dissatisfaction for every student group k, professor i and time slot j triple, find a scheduling that minimizes the total (maximal) dissatisfaction.

Balas and Landweer [9] used the P3AP to model satellite launching. Furthermore, the P3AP is used to model timetabling (see e.g. [18, 48]) and rostering problems [42].

2.3.2 Complexity and algorithms

The P3AP and the p-P3AP are NP-hard. A feasible solution of the p-P3AP is made of p disjoint permutations, i.e. p disjoint perfect bipartite matchings. Frieze [39] proved that already the problem concerning only two disjoint matchings is hard, i.e. the Disjoint Matchings is NP-complete.

> Disjoint Matchings (DM)
> Input: Disjoint finite sets P, Q with $|P| = |Q|$ and sets $A_1, A_2 \subseteq P \times Q$.
> Question: Do there exist perfect matchings $M_1 \subseteq A_1$ and $M_2 \subseteq A_2$ such that $M_1 \cap M_2 = \emptyset$?

A simple proof of Frieze's result can be found in [37].

Given an instance of the DM we can define an $n \times n \times p$ array $C = (c_{ijk})$ as follows. Let elements of sets P and Q correspond to indices i and j, respectively. For $k = 1, 2$ let c_{ijk} be 0 if the pair that corresponds to (i, j) is in A_k, and 1 otherwise. For $k \geq 3$ let $c_{ijk} = 0$. Then, given DM instance is a YES instance if and only if the objective value of an optimal solution of the (bottleneck) p-P3AP with the cost array C is 0. This is true because every pair of two disjoint perfect bipartite matchings can be extended to the set of $p \leq n$ disjoint perfect bipartite matchings, see [46]. For the max-p-P3AP swap 0's and 1's in C. Hence, the (bottleneck) p-P3AP is NP-hard even when restricted to 0-1 cost arrays. Moreover, it is straightforward to see that for the p-P3AP and bottleneck-p-P3AP no polynomial algorithm can ever achieve a constant performance ratio unless P=NP. Conversely, using the result from [8] a $\frac{1}{2}$-approximation algorithm can be obtained for the max-P3AP version with inequalities ≤ 1 in (2.18), (2.19) and (2.20).

Branch and bound methods for the P3AP are presented by Vlach [79] and Magos and Miliotis [59]. A tabu search approach is presented in [58]. The polyhedral structure related to the P3AP is studied in [5, 35, 36]. Frieze and Sorkin [40] have established high probability lower and upper bounds for the optimal objective function value under the assumption that costs c_{ijk} are independent, identically distributed exponential random variables with parameter 1.

The p-P3AP has received less attention in the literature than the P3AP. Gimadi and Glazkov [44] suggested an approximation algorithm for the p-P3AP and analysed its performance in the case when the costs are independent, identically distributed uniform random variables. Yokota et al. [80] presented an algorithm for the p-P3AP (called repeated assignment problem in their paper) that outperforms general ILP-solvers for instances with small p.

2.3.3 Special cases

We are not aware of any results in the literature on non-trivial special cases of the P3AP. However, several contributions to this topic are presented in this thesis.

In Chapter 4 special cases that satisfy multiple variants of Monge properties are investigated for the P3AP together with a variation of the P3AP for which a feasible solution consists of p disjoint perfect matchings, $2 \leq p \leq n$.

In Chapter 5 we characterized all special cases of the P3AP (together with other problems and generalizations) for which every feasible solution has the same objective value.

2.4 Multidimensional assignment problems

In previous sections we defined classical two-dimensional and three-dimensional assignment problems, namely the LAP, the A3AP and the P3AP. In this section we describe how the notion of assignment problems can naturally be extended to higher dimensions. Characterizations of the multidimensional assignment problem cost arrays for which every feasible solution has the same objective value are investigated in Chapter 5.

2.4.1 Definition

In the case of the LAP we are given a two-dimensional $n \times n$ cost matrix, and every feasible solution corresponds to a set of n cost elements, one for every one fixed index (row or column) of the cost matrix. In the case of the A3AP and the P3AP we are given a three-dimensional $n \times n \times n$ cost array. Every feasible solution of the A3AP corresponds to a set of n cost elements, one for every one index of the cost array fixed. In the case of the P3AP every feasible solution corresponds to a set of n^2 cost elements, one for every two indices of the cost array fixed. Now it is straightforward to generalize these notions to higher dimensions. A general multidimensional assignment problem is specified by two parameters d and s, where d is the number of indices and s describes the number of fixed indices in the constraints. Informally speaking, we want to find a set of n^s elements of a d-dimensional $n \times n \times \cdots \times n$ array C with minimal total sum, such that for every s fixed indices of C exactly one element is chosen.

Formally, the (d, s) assignment problem, (d, s)-AP for short, can be stated in the following way.

Definition 2.5. Let d and s be integers with $0 < s < d$. The input of the (d, s)-AP consists of an integer $n \geq 1$ and a d-dimensional $n \times n \times \cdots \times n$ cost array C which associates the cost $c(i_1, i_2, \ldots, i_d)$ to the d-tuple $(i_1, i_2, \ldots, i_d) \in \{1, \ldots, n\}^d$. Let $\mathcal{Q}_s$ be the set of all subsets of

$K = \{1, \dots, d\}$ with cardinality s, i.e. $\mathcal{Q}_s = \{Q \colon Q \subset K, |Q| = s\}$. For any set $Q = \{q_1, q_2, \dots, q_s\} \in \mathcal{Q}_s$ of fixed indices with $q_1 < q_2 < \dots < q_s$ and any s-tuple $t = (t_1, \dots, t_s) \in \{1, \dots, n\}^s$, let $T(Q, t)$ be the set of all d-tuples $t' = (t'_1, \dots, t'_d) \in \{1, \dots, n\}^d$ such that $t'_{q_j} = t_j$ for all $j = 1, \dots, s$. The general (d, s) assignment problem (d, s)-AP can be stated as

$$\min \sum_{i_1=1}^{n} \cdots \sum_{i_d=1}^{n} c(i_1, i_2, \dots, i_d) x(i_1, i_2, \dots, i_d) \tag{2.22}$$

$$\begin{aligned} \text{s.t.} \quad \sum_{\substack{(i_1,\dots,i_d)\in \\ T(Q,(j_1,\dots,j_s))}} x(i_1, i_2, \dots, i_d) = 1 \quad & \text{for all } Q \in \mathcal{Q}_s \\ & \text{and all } (j_1, \dots, j_s) \in \{1, \dots, n\}^s, \\ x(i_1, i_2, \dots, i_d) \in \{0, 1\} \quad & \text{for all } (i_1, i_2, \dots, i_d) \in \{1, \dots, n\}^d. \end{aligned}$$

Note that in each of the equality constraints above the sum essentially extends over $d - s$ variables (corresponding to the free indices from the set $K \setminus Q$).

The *bottleneck (d, s) assignment problem* (bottleneck-(d, s)-AP) is the version of the problem where the maximal element is minimized, i.e. we have

$$\min \max_{i_1=1,\dots,n} \cdots \max_{i_d=1,\dots,n} c(i_1, i_2, \dots, i_d) x(i_1, i_2, \dots, i_d)$$

instead of (2.22).

Let $X = (x(i_1, i_2, \dots, i_d))$ be a feasible solution of the integer program stated above. Then the set $F = \{(i_1, i_2, \dots, i_d) \colon x(i_1, i_2, \dots, i_d) = 1\}$ is a feasible solution of the (d, s)-AP and the value $\sum_{(i_1,\dots,i_d)\in F} c(i_1, \dots, i_d)$ is the cost (objective value) of the feasible solution F.

Using the (d, s)-AP notation, the LAP is the $(2, 1)$-AP, while the A3AP and the P3AP correspond to the $(3, 1)$-AP and the $(3, 2)$-AP, respectively. More generally, we refer to the $(d, 1)$-AP as *axial d-dimensional assignment problem*, and to the $(d, d-1)$-AP as *planar d-dimensional assignment problem.*

Let us remark that for $d \geq 4$, there is no consensus in the literature which problem version is referred to as planar d-dimensional assignment problem. Our decision to refer to the $(d, d-1)$-AP in which all constraints involve single sums as planar problem is in accordance with the axial/planar nomenclature which has been introduced in the original paper by Schell [74] and is used by Spieksma [76], Frieze and Sorkin [40] and others. A group of authors around Appa, see e.g. [5], refer to the $(d, 2)$-AP as planar problem.

The A3AP and the P3AP are known to be NP-hard [39, 52]. As a consequence thereof both the axial and planar d-dimensional assignment problems are NP-hard for all $d \geq 3$.

2.4.2 Feasible solutions

The following observations collect a few well known facts about the structure of the set of feasible solutions of the (d, s)-AP.

Observation 2.6. Every feasible solution of the $(d, 1)$-AP of size n can be represented by a set of $d-1$ permutations of $\{1, \dots, n\}$. Namely, every feasible solution can be written as $F = \{(i, \phi_1(i), \dots, \phi_{d-1}(i)) \colon i = 1, \dots, n\}$ where each ϕ_k is a permutation of $\{1, \dots, n\}$.

Observation 2.7. A set of d-tuples F is a feasible solution of the $(d, d-1)$-AP of size n if and only if F "contains" n pairwise disjoint feasible solutions of the $(d-1, d-2)$-AP. Namely, define n sets of $(d-1)$-tuples obtained from F by fixing one index, for example the first one: $F_i = \{(a_2, a_3, \dots, a_d) \colon (i, a_2, a_3, \dots, a_d) \in F\}$ for $i = 1, 2, \dots, n$. Then every F_i is a feasible solution of the $(d-1, d-2)$-AP because fixing $d-2$ indices in F_i corresponds to fixing $d-1$ indices in F. Also, if there are F_i and F_j that are not disjoint, then there would be two elements in F that coincide on $d-1$ indices, a contradiction. The same construction works in the other direction as well.

Observation 2.8. The feasible solutions of the $(d, 2)$-AP for $d \geq 3$ correspond to $(d-2)$-tuples of mutually orthogonal Latin squares. Indeed, assume that F is a feasible solution of the $(d, 2)$-AP. We can represent F as an $n \times n$ table T with $(d-2)$-tuples as entries in the following way: The $(d-2)$-tuple $(i_3, \dots, i_d)$ is the entry in row i_1 and column i_2 of T if and only if $(i_1, \dots, i_d)$ is an element of F. Since F is a feasible solution of the $(d, 2)$-AP, each row and each column of T contain every integer from 1 to n exactly once on the k-position for all $k = 1, \dots, d-2$. Moreover, each $(d-2)$-tuple of pairwise distinct integers from $\{1, \dots, n\}$ appears exactly once in T. Hence T can be interpreted as a $(d-2)$-tuple of mutually orthogonal Latin squares (the k-th component of the entries of T yields the k-th Latin square). Note that if $d = 4$ then T is a Graeco-Latin square.

What distinguishes the general (d, s)-AP from the special cases with $s = 1$ (axial problem) and with $s = d - 1$ (planar problem) is that there does not need to exist feasible solutions for every value of n. Furthermore, not much is known on the structure of the set of feasible solutions for the (d, s)-AP for general n, cf. Appa et al. [6]. Infeasible instances and instances with very few feasible solutions provide clear obstacles to our intended COVP characterization. For this reason the feasibility topic for the (d, s)-AP plays a role for us and we briefly review a few basic results from the literature.

The question for which values n the general (d, s)-AP has feasible solutions is a very difficult problem which is still open for many combinations of d and s, and is related to a number of difficult problems in combinatorics. More precisely, the general (d, s)-AP of size n has a feasible solution if there exists an s-transversal design with d groups of size n, or equivalently if there exists an orthogonal array $\mathrm{OA}(n, d, s)$ of index 1, strength s and order n, see [25].

In view of Observation 2.8, the question of existence of Graeco-Latin squares is of interest. This was a famous open problem for a long time going back to Euler until it was finally proved by Bose et al. [17] that Graeco-Latin squares, and thus feasible solutions of the $(4, 2)$-AP, exist for every $n \geq 3$ except for $n = 6$.

The number of mutually orthogonal Latin squares of size n (see Observation 2.8 for the connection to the feasibility of the $(d, 2)$-AP) is unknown for general n. It is known however that this number is at most $n - 1$ and that the upper bound is achieved if n is a prime power. It is also known that there exist $n - 1$ mutually orthogonal Latin squares if and only if there exists a projective plane of order n, see [25].

2.4.3 Applications and algorithms

An important application that is modeled using an axial d-dimensional assignment problem occurs in the field of target tracking and plays a central role in operating surveillance systems. Solution methods for this application based on Lagrangian relaxation are proposed in Pattipati et al. [61], Poore [67], Poore and Rijavec [68]. Other approaches are described in Murphey et al. [60].

An application concerning routing in meshes that can be formulated as the $(d, 1)$-AP is described in Fortin and Tusera [38]. The data association problem, stemming from high energy physics and leading to the $(5, 1)$-AP is described by Pusztaszeri, Rensing, and Liebling [70] (see also Pusztaszeri [69]). The authors tried to reconstruct the tracks of charged elementary particles generated by the large electron-positron collider at CERN in Geneva.

The $(4, 2)$-AP occurs in the design of tournaments (see Appa, Magos, and Mourtos [4] for a branch and cut approach) and conflict-free access to parallel memories and the design of error-correcting codes.

Chapter 3

Geometric axial 3-dimensional assignment problems

The axial 3-dimensional assignment problem (A3AP) is an important and well-studied problem in combinatorial optimization, see Section 2.2. An instance of the A3AP consists of three sets X, Y, Z with $|X| = |Y| = |Z| = n$, and a cost function $c : X \times Y \times Z \to \mathbb{R}$. The goal is to find a set of n triples in $X \times Y \times Z$ that cover every element in $X \cup Y \cup Z$ exactly once, so that the sum of the costs of these triples is minimized. In the closely related maximization version max-A3AP of the A3AP, this sum is to be maximized. The book by Burkard et al. [21] contains a wealth of information on the A3AP and other assignment problems.

A prominent special case of the A3AP is centered around a metric space (S, d) where S is a set and where d is a distance function on S (that hence is symmetric, non-negative, and satisfies the triangle inequality). The elements in $X \cup Y \cup Z$ are points in S, and the cost $c(x, y, z)$ of a triple $(x, y, z) \in X \times Y \times Z$ is given by

$$c(x, y, z) \; = \; d(x, y) + d(y, z) + d(z, x). \tag{3.1}$$

Costs of this type are sometimes called *perimeter costs*; intuitively speaking, they measure the perimeter of the triangle determined by points x, y, z in the metric space. Investigations under these types of costs are the topic of this chapter. The results presented here can be found in [29].

3.1 Technical preliminaries

Let R denote a compact and convex subset of the s-dimensional Cartesian space $\mathbb{R}^s$ that has non-empty interior and that is centrally symmetric with respect to the origin. The corresponding norm L_R with unit ball R determines for any two points $x, y \in \mathbb{R}^s$ a distance $d_R(x, y)$ in the following way. First translate the space so that one of the two points (say point x) lies in the origin. Then determine the unique scaling factor λ by which one must rescale the unit ball R (shrinking for $\lambda < 1$, expanding for $\lambda > 1$), so that the other point (point y, in our case) lies on its boundary. The distance is then given by $d_R(x, y) = \lambda$. Note that since R is centrally symmetric, it does not matter whether we choose point x or point y for the origin. See Figure 3.1 for an illustration.

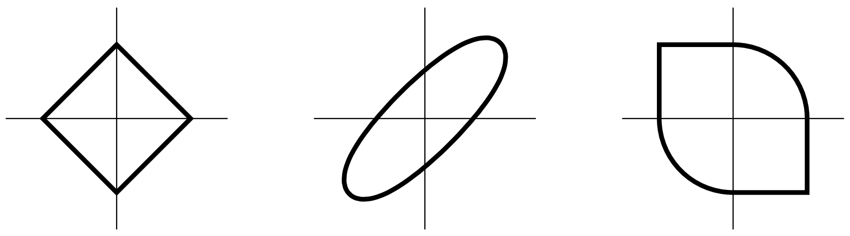
Figure 3.1: Three examples of unit balls for an L_R norm in $\mathbb{R}^2$

The most popular norms for $\mathbb{R}^s$ are the Manhattan norm, the Euclidean norm, and the Maximum norm. These three norms are special cases of the well-known L_p norm, respectively for $p = 1$, for $p = 2$, and for $p = \infty$. We recall that for $1 \le p < \infty$, the L_p distance between two points $x = (x_1, \ldots, x_s)$ and $y = (y_1, \ldots, y_s)$ in s-dimensional space is given by

$$d(x, y) \;=\; \left(\sum_{i=1}^{s} |x_i - y_i|^p \right)^{1/p} . \tag{3.2}$$

For $p = \infty$, the corresponding distance under the Maximum norm L_∞ is given by

$$d(x, y) \;=\; \max{}_{i=1}^{s} |x_i - y_i|. \tag{3.3}$$

3.2 The maximization problem under tunneling distances

In this section we consider a variant of the max-A3AP with perimeter costs that will be useful in Section 3.3. The distances between the elements of $X \cup Y \cup Z$ are specified with the help of a system of $k \geq 2$ so-called *tunnels* $t_1, \ldots, t_k$; we stress that throughout this section the number k of tunnels is a constant that does not depend on the input. Each tunnel acts as a bidirectional passage with a front entry and a back entry. For every element $x \in X \cup Y \cup Z$ and every tunnel t, we denote by $F(x,t)$ the distance between x and the front entry of t and by $B(x,t)$ the distance between x and the back entry of t. Intuitively speaking, the only way of moving from x to y is to first move from x to some tunnel, then to traverse the tunnel in either direction (either from front entry to back entry, or from back entry to front entry), and finally to move from the other end of the tunnel to y. The *tunneling distance* between two elements x and y in $X \cup Y \cup Z$ is then given by

$$d(x,y) = \max\left\{F(x,t_i) + B(y,t_i),\ B(x,t_i) + F(y,t_i) \colon 1 \leq i \leq k\right\}. \quad (3.4)$$

The maximization version of the traveling salesman problem under tunneling distances is studied in [12].

We construct an undirected, edge-labeled, bipartite multigraph G whose vertex set are the elements of $X \cup Y \cup Z$ together with the tunnels $t_1, \ldots, t_k$. Between any element x of $X \cup Y \cup Z$ and any tunnel t there are two edges, one of which is labeled B and has cost $B(x,t)$, whereas the other one is labeled F and has cost $F(x,t)$.

A *six-cycle* in G is a closed walk $x - t_i - y - t_j - z - t_k - x$ with $(x,y,z) \in X \times Y \times Z$ and three (not necessarily distinct) tunnels t_i, t_j, t_k. The six-cycle is *legal*, if the labels of the two edges incident to t_i are distinct, if the labels of the two edges incident to t_j are distinct, and if the labels of the two edges incident to t_k are distinct. We stress that we do not require tunnel t_i to be the maximizer of the expression for x and y in the right hand side of (3.4), nor that t_j and t_k are the maximizers for the corresponding expressions for y and z, respectively for z and x.

A *legal* set C of six-cycles consists of n legal six-cycles in G that cover every vertex of $X \cup Y \cup Z$ exactly once. We define $G[C]$ as the subgraph of G that is induced by the $6n$ edges in C. Then we coarsen the subgraph $G[C]$ by anonymizing the identities of the vertices in $X \cup Y \cup Z$: every vertex in X is simply labeled X, every vertex in Y is labeled Y, and every vertex in Z is labeled Z. The resulting anonymized graph $G^*[C]$ is called an *outline* of C and $G[C]$.

Lemma 3.1. *The optimal objective value of the considered max-A3AP instance coincides with the largest cost taken over all subgraphs $G[C]$ with a legal set C of six-cycles.*

Proof. Let C be an arbitrary legal set of six-cycles. Every six-cycle $x - t_i - y - t_j - z - t_k - x$ in C yields a corresponding triple (x, y, z) in $X \times Y \times Z$. The cost $c(x, y, z)$ of triple (x, y, z) may be computed according to (3.4), by replacing the three tunnels t_i, t_j, t_k by three other tunnels that maximize the value. Hence, the cost of the triple is an upper bound on the cost of the six-cycle, and the cost of all n corresponding triples is an upper bound on the cost of $G[C]$. This shows that the optimal objective max-A3AP value is an upper bound on the cost of every subgraph $G[C]$.

Next consider a set T of n triples in $X \times Y \times Z$ that constitutes an optimal solution for the max-A3AP instance. We translate every triple $(x, y, z) \in T$ into a legal six-cycle: we let t_i (t_j and t_k, respectively) denote the tunnel that maximizes the expression (3.4) for x and y (respectively, for y and z and for z and x), and we choose the labels B and F appropriately in the obvious way. For the resulting legal set C_T, the cost of $G[C_T]$ coincides with the optimal objective max-A3AP value. $\square$

Lemma 3.2. *Let G^* be a given outline. Then we can compute in polynomial time $O(n^3)$ the largest cost of all the induced subgraphs $G[C]$ (with a legal set C of six-cycles), whose outline $G^*[C]$ coincides with G^*.*

Proof. The problem boils down to assigning the elements of X (respectively, of Y and Z) to the n vertices in G^* that are labeled X (respectively, labeled Y and Z). The cost of assigning an element $x \in X$ to some vertex v only depends on x and on the two edges incident to v in G^*. Hence, we are dealing with a classical two-dimensional assignment problem which can be solved in polynomial time $O(n^3)$; see for instance [21]. $\square$

Lemma 3.3. *There exist only $O(n^{8k^3})$ distinct outlines for graph G, and they can all be enumerated in polynomial time.*

Proof. After anonymizing the identities of the vertices in $X \cup Y \cup Z$, a legal six-cycle is determined by the three tunnels t_i, t_j, t_k and the labels of its first, third, and fifth edge. Hence there remain only $8k^3$ combinatorially different legal six-cycles, and each of them may be used at most n times in any outline. □

The three lemmas above suggest the following approach to the max-A3AP under tunneling distances. Enumerate all possible outlines in polynomial time according to Lemma 3.3, and for each such outline compute the maximum possible cost of a corresponding induced subgraph according to Lemma 3.2. Return the largest cost over all outlines, which by Lemma 3.1 coincides with the optimal objective value of the max-A3AP instance.

Theorem 3.4. *Problem max-A3AP with perimeter costs under tunneling distances can be solved within a time complexity that depends polynomially on the instance size n (and exponentially on the number k of tunnels).*

The approach above computes the optimal objective value and the corresponding graphs $G^*[C]$ and $G[C]$, but does neither yield the corresponding optimal solution $T \subset X \times Y \times Z$ for the max-A3AP instance, nor the underlying legal set C_T of six-cycles. The set C_T can be determined in polynomial time by invoking Lenstra's algorithm [56] for integer programming in constant dimension. For each of the $8k^3$ combinatorially different legal six-cycles, we introduce a corresponding integer variable that counts the number of occurrences of this cyle in C_T. The constraints in the integer program enforce that $G[C_T]$ coincides with $G[C]$. And once we have found C_T via the integer program, it is straightforward to identify the optimal solution T (as outlined in the proof of Lemma 3.1).

3.3 The maximization problem under polyhedral norms

Throughout this section, we consider the s-dimensional Cartesian space $\mathbb{R}^s$ endowed with some fixed norm with polyhedral unit ball R. We investigate the special case of max-A3AP with perimeter costs where the elements in $X \cup Y \cup Z$ are points in $\mathbb{R}^s$ and where the distances are measured according to d_R. We stress that both the dimension s of the underlying space and the number of faces of the unit ball R are constants that do not depend on the input.

The unit ball R is a polytope with $2k$ faces that is centrally symmetric with respect to the origin. Then for certain vectors $h_1, \ldots, h_k \in \mathbb{R}^s$, this polytope R can be written as the intersection of a collection of half-spaces:

$$R = \left(\bigcap_{i=1}^{k} \{x : \; h_i \cdot x \leq 1\} \right) \cap \left(\bigcap_{i=1}^{k} \{x : \; h_i \cdot x \geq -1\} \right) \tag{3.5}$$

As an example, for the rectilinear (Manhattan) norm in $\mathbb{R}^2$ the corresponding vectors are $h_1 = (1,1)$ and $h_2 = (-1,-1)$, and for the maximum norm in $\mathbb{R}^2$ the corresponding vectors are $h_1 = (1,0)$ and $h_2 = (0,1)$. The distance $d_R(x,y)$ between two points $x, y \in \mathbb{R}^s$ is then given by

$$\begin{aligned} d_R(x,y) &= \max \{ |h_i \cdot (x-y)| : \; 1 \leq i \leq s \} \\ &= \max \{ h_i \cdot (x-y), \; h_i \cdot (y-x) : \; 1 \leq i \leq s \} \\ &= \max \{ h_i \cdot x - h_i \cdot y, \; -h_i \cdot x + h_i \cdot y : \; 1 \leq i \leq s \} . \end{aligned} \tag{3.6}$$

We model a max-A3AP instance under a polyhedral norm as a special max-A3AP under tunneling distances as discussed in Section 3.2. The k vectors $h_1, \ldots, h_k$ serve as tunnels, and we set $F(x, h_i) = x \cdot h_i$ and $B(x, h_i) = -x \cdot h_i$. With this choice, the polyhedral distance $d_R(x,y)$ between two points x and y in $X \cup Y \cup Z$ in (3.6) coincides with the distance given in (3.4). Hence Theorem 3.4 yields the following.

Theorem 3.5. *For any polyhedral norm L_R with unit ball R in s-dimensional space $\mathbb{R}^s$, the max-A3AP with perimeter costs measured according to L_R can be solved within a time complexity that depends polynomially on the instance size n (and exponentially on the number k of facets of the polyhedral unit ball).*

Theorem 3.5 implies the existence of a polynomial time approximation scheme (PTAS) for the max-A3AP under any arbitrary norm with a not necessarily polyhedral unit ball R. One simply approximates the unit ball R by a polyhedral unit ball. Since the dimension s of the underlying space and the ball R are fixed, one may choose a fixed polyhedral approximation of the ball that approximates the distances between any two points within a factor $1 \pm \varepsilon$. (This trick of approximating the unit ball is due to Barvinok [13] who applied it to the maximum traveling salesman problem.) Hence the following theorem holds.

Theorem 3.6. *For any* (*not necessarily polyhedral*) *norm L_R with unit ball R in s-dimensional space $\mathbb{R}^s$, the max-A3AP with perimeter costs measured according to L_R possesses a PTAS.*

3.4 The maximization problem in non-fixed dimension

The polynomial time results for the max-A3AP in the preceding section assumed that the dimension s of the underlying space and the number of faces of the underlying unit ball are constants that do not depend on the input. In this section we discuss problem max-A3AP with perimeter costs measured according to a standard L_p norm (with $1 \leq p \leq \infty$) if the dimension s is not fixed and part of the input. Our reductions are from the following variant of Partition into Triangles.

> Partition into Triangles (PIT)
>
> Instance: A 6-regular, tripartite graph $G = (V, E)$ with tripartition $V = V_1 \cup V_2 \cup V_3$, where $|V_1| = |V_2| = |V_3| = q$.
>
> Question: Does there exist a set T of q triples in $V_1 \times V_2 \times V_3$ such that every vertex in V occurs in exactly one triple and such that every triple induces a triangle in G?

We have not been able to locate an NP-hardness proof of PIT on 6-regular tripartite graphs in the literature (though we strongly expect that this result has been observed before). For instance Van Rooij et al. [78] establish NP-hardness for 4-regular graphs, but their graphs are not tripartite.

Proposition 3.7. *The PIT on* 6*-regular tripartite graphs is NP-complete.*

Proof. The argument is routine, and we only sketch the main ideas. The NP-hardness proof on pages 68 and 69 of Garey and Johnson [41] for Partition into Triangles is a reduction from the Exact Cover By 3-Sets problem. We perform essentially the same reduction, but start it from the feasibility version of the axial 3-dimensional assignment problem (Instance: three sets X, Y, Z with $|X| = |Y| = |Z| = q$, and a set $T \subseteq X \times Y \times Z$ of triples such that every element of $X \cup Y \cup Z$ occurs in at most three triples of T. Question: Does there exist a subset T^* of q triples in T such that each element of $X \cup Y \cup Z$ is contained in precisely one triple of T^*?) Then the resulting graph G is tripartite and all vertex degrees lie in $\{3, 4, 5, 6\}$.

Hence it remains to make the graph 6-regular. This can be reached by various gadget constructions. We sketch a particularly simple approach that increases the minimum degree of G by one, while keeping the maximum degree unchanged. Take the graph $G = (V, E)$, and construct a copy $G' = (V', E')$ of it (so that for every $v \in V$ there is a corresponding copy $v' \in V'$, and there is an edge $(u, v) \in E$ if and only if there is an edge $(u', v') \in E'$). Define a new graph on the vertex set $V \cup V'$, and all edges in $E \cup E'$, and furthermore an additional edge between v and v' whenever vertex v has degree in $\{3, 4, 5\}$. The new graph is still tripartite, and it has a partition into triangles if and only if the old graph allows a partition into triangles (note that the additional edges (v, v') do not occur in any triangle, and hence are irrelevant). If we repeat this construction two more times, the resulting graph will be 6-regular and tripartite. $\square$

The following two lemmas establish NP-hardness of max-A3AP with perimeter costs for all values p with $1 \le p \le \infty$.

Lemma 3.8. *For any fixed p with $1 \le p < \infty$, the max-A3AP with perimeter costs measured according to the L_p norm is NP-hard.*

Proof. We consider an arbitrary instance $G = (V, E)$ of PIT, and we construct the following instance of max-A3AP with perimeter costs from it. For every vertex v in part V_1 (respectively, part V_2 and part V_3), we create a corresponding point $P(v)$ that belongs to the set X (respectively, set Y and set Z). We choose $s = \binom{n}{2}$, and we make every coordinate correspond to one 2-element set of vertices in V. The coordinate of point $P(v)$ corresponding to some set $\{u, w\}$ with $u, w \in V$ is chosen as follows: If

$v \in \{u, w\}$ and (u, w) is not an edge in E, then the coordinate has value 1; in all other cases the coordinate has value 0.

Since G is 6-regular, every vertex v has exactly $3q-7$ non-neighbors and every point $P(v)$ has exactly $3q-7$ coordinates with value 1 (and all other coordinates at 0). Furthermore, if $(u, v) \in E$ then the L_p distance between $P(u)$ and $P(v)$ equals $\ell^* := \sqrt[p]{6q-14}$, and if $(u, v) \notin E$ then their L_p distance equals $\sqrt[p]{6q-16}$. In other words, non-edges correspond to short distances and edges correspond to long distances. It can be seen that the PIT instance has answer YES, if and only if the constructed max-A3AP instance has a feasible solution with objective value at least $3q \cdot \ell^*$. $\square$

Lemma 3.9. *The max-A3AP with perimeter costs measured according to the Maximum norm L_∞ is NP-hard.*

Proof. The argument is very similar to the argument in Lemma 3.8. Again we start from an arbitrary instance $G = (V, E)$ of PIT, and we create for every vertex v in $V_1 \cup V_2 \cup V_3$ a corresponding point $P(v)$. We choose $s = |E|$, and we make every coordinate correspond to one edge in E. For an edge $e = (u, v) \in E$, the coordinates corresponding to e are 0 for all points with the exception of $P(u)$ and $P(v)$; one of $P(u)$ and $P(v)$ receives coordinate $+1$ and the other one receives coordinate -1.

Then non-edges correspond to short distances 1 and edges correspond to long distances $\ell^* := 2$. It can be seen that the PIT instance has answer YES, if and only if the constructed max-A3AP instance has a feasible solution with objective value at least $6q$. $\square$

3.5 A useful lattice

In this section, we derive a technical result (Theorem 3.12) that will be central in the NP-hardness construction in Section 3.6. Throughout the section we consider a fixed norm L_R with unit ball R in the Cartesian plane $\mathbb{R}^2$.

In our NP-hardness reduction we place generated points onto a 2-dimensional lattice. Our lattice will be defined by a fundamental triangle $p_1p_2p_3$ whose shape and size depend on R, and is constructed as follows.

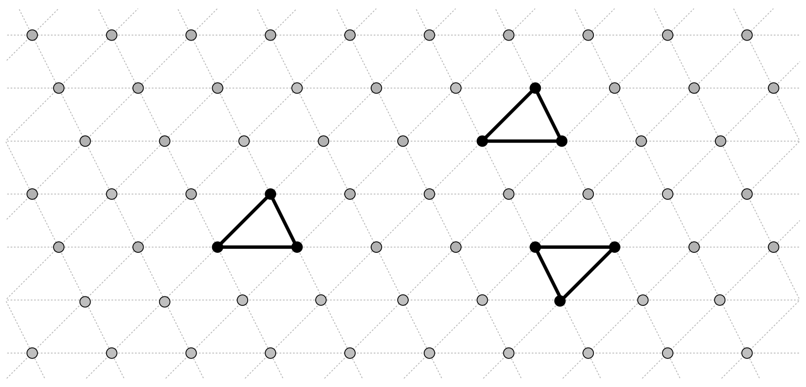

Figure 3.2: Lattice and its fundamental triangles

We start by setting p_1 to be the origin and p_2 to be the point $(1, 0)$. Later, in the process of resizing, the coordinates of p_2 could be changed. Let $d_R(p, q)$ denote the L_R distance of points p and q as before. We will choose the third point p_3 from the region A, which is defined as the set of points in the upper half-plane of $\mathbb{R}^2$ with L_R distance from both p_1 and p_2 strictly greater than $d_R(p_1, p_2)$ and less or equal than $1.2d_R(p_1, p_2)$, and such that all points in A have L_R distance from points $p_L := (-1, 0)$ and $p_R := (2, 0)$ strictly greater than $1.2d_R(p_1, p_2)$, see Figure 3.3. In

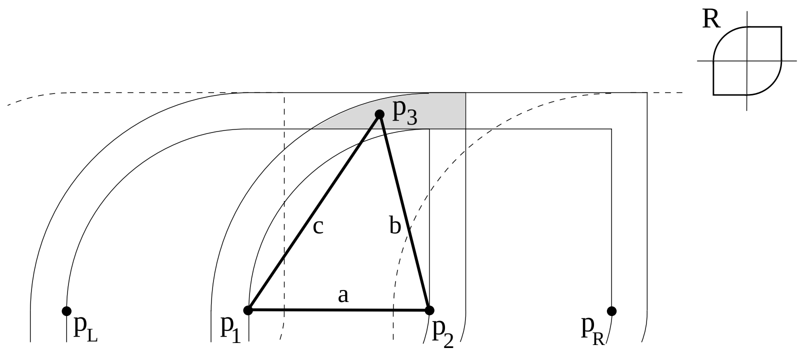

Figure 3.3: Fundamental triangle construction example

Lemma 3.10 the existence of such an area for every R will be shown. We choose p_3 to be any point from A with integer coordinates. If there is no such point in A we multiply all coordinates by some integer so that such a point appears. We denote the side sizes of the triangle $p_1p_2p_3$ by $a := d_R(p_1, p_2)$, $b := d_R(p_2, p_3)$ and $c := d_R(p_1, p_3)$. We further denote $di_L := d_R(p_L, p_3)$ and $di_R := d_R(p_R, p_3)$. Lastly we inflate the triangle $p_1p_2p_3$ by multiplying all coordinates by some integer so that the condition

$\min\{di_L - b, di_R - c, \max\{b, c\} - a\} \geq 1$ is satisfied (this will ensure that the difference between the perimeter of the fundamental triangle and any non-fundamental triangle is at least 1, see Lemma 3.11). This completes the construction of our fundamental triangle.

We denote the perimeter (in L_R norm) of the triangle $p_1p_2p_3$ by Δ. Lattice defining lines (dotted lines in Figure 3.2) fall into three classes; the horizontal ones (parallel to p_1p_2) we call a-lines, those parallel to p_2p_3 we call b-lines, and those parallel to p_1p_3 we call c-lines. Furthermore, in the proof of Lemma 3.11 we will make use of lines parallel to p_3p_L and p_3p_R, which we denote as *right and left diagonal lines*, respectively.

Lemma 3.10. *Given any L_R norm, the set A defined above is a region with nonempty area.*

Proof. Let us denote by S the set of points in the upper half-plane whose L_R distance from both p_1 and p_2 is exactly $1.2a$. We claim that the set S will consist either of only one point or of one horizontal segment. Assume that this is not the case, i.e. we have two points in S which are not on the same horizontal line. The lower one we denote by s_1 and the upper one by s_2. Then by shifting s_1 and s_2 to the right by a we get points r_1 and r_2, respectively, which are two additional points whose L_R distance from p_2 is exactly $1.2a$. Now consider the line that passes through points p_2 and r_1. We distinguish three cases, depending on the position of point r_2 with respect to the line p_2r_1.

Case I: Line p_2r_1 passes through point r_2. In that case we have two distinct points r_1 and r_2 that have the same distance from point p_2 in the same direction; contradiction.

Case II: Line p_2r_1 is positioned to the left of point r_2. Denote by x the intersection of lines p_2r_1 and s_1r_2, see Figure 3.4. The convexity of R implies that $d_R(p_2, x) \leq 1.2a$, which is in contradiction with the fact that x is further away from p_2 than r_1.

Case III: Line p_2r_1 is positioned to the right of point r_2. Denote by q the point on line p_1p_2 which is to the right of p_2 and is such that $d_R(p_2, q) = 1.2a$. We consider two subcases. In the first subcase, lines r_2r_1 and p_1p_2 intersect to the left of point q, see Figure 3.5a. Then the point of intersection of lines r_2q and p_2r_1 and the point r_1 imply a con-

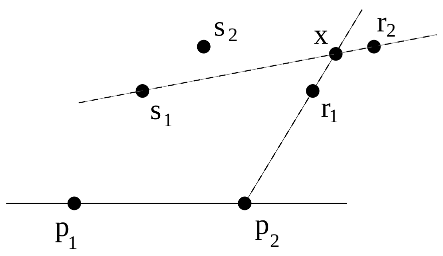

Figure 3.4: Instance of Case II.

tradiction in the same manner as in Case II. In the second subcase, lines r_2r_1 and p_1p_2 intersect at point q or to the right of it, see Figure 3.5b. In that case, lines s_2s_1 and p_1p_2 intersect to the right of p_2. Then the point of intersection of lines p_2s_1 and s_2r_1 and the point s_1 imply a contradiction in the same manner as in Case II.

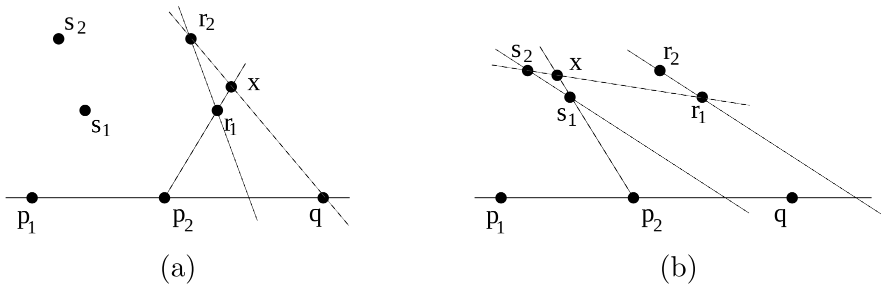

Figure 3.5: Instances of Case III.

We obtained a contradiction in every case, hence all points in S are on one horizontal line.

Next we prove that for any two points s_1, s_2 from S the whole segment $\overline{s_1s_2}$ is in S. Assume that $\overline{s_1s_2}$ is not on the border of R scaled by $1.2a$ and centered in p_2. Then there exists a point q such that $d_R(p_2, q) = 1.2a$ with x-coordinate greater than that of s_1 and smaller than that of s_2, and y-coordinate greater than that of s_1 and s_2. Assuming that s_2 is to the right of s_1, let r be the point obtained by shifting s_2 to the right by a. Then the line p_2s_2 intersects either segment $\overline{s_1q}$ or segment $\overline{qr}$ at the point which we denote by x (see Figure 3.6), which leads to a contradiction in the same manner as before. This shows that S contains one point or one horizontal segment.

Finally, we prove that there exists a point in S whose neighborhood contains a region which is in A. Denote the left and the right endpoint of

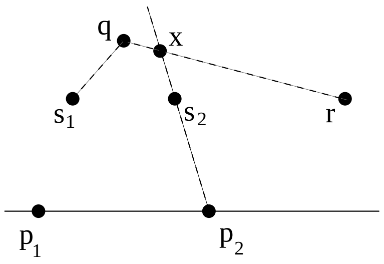

Figure 3.6: S must be a horizontal segment

a segment S by s_l and s_r, respectively. Denote by q the point obtained by shifting s_l to the left by a, and denote by p_R (p_L) the point obtained by shifting p_2 (p_1) to the right (left) by a. Note that $d_R(p_1, q) = d_R(p_2, s_l) = 1.2a$. Then $d_R(p_2, q) = d_R(p_R, s_l) > 1.2a$. Indeed, if $d_R(p_2, q) = 1.2a$, then q is in S, and if $d_R(p_2, q) < 1.2a$, then denote by x the intersection of $p_2 s_l$ and $q_2 s_r$, where q_2 is on the line $p_2 q$ such that $d_R(p_2 q_2) = 1.2a$; point x leads to a contradiction in the same manner as in Case I-III above. Analogously it follows that $d_R(p_L, s_r) > 1.2a$. Let $S_L := \{x \in S \colon d_R(p_L, x) \leq 1.2a\}$ and analogously $S_R := \{x \in S \colon d_R(p_R, x) \leq 1.2a\}$. It follows that both S_L and S_R are proper subsets of S. Let $s \in S$ be a point such that $d_R(p_L, s) > 1.2a$ and $d_R(p_R, s) > 1.2a$. Note that such a point exists. Namely, otherwise we have that $S_L \cup S_R = S$ and the intersection of closures of S_L and S_R is not empty, and any point from this intersection together with points p_L and p_R does not satisfy the triangle inequality ($1.2a + 1.2a \geq 3a$), contradiction. Hence such point s exists.

Finally we show that a neighborhood of s contains an area that is in A. Let B_ϵ be a ball around the point s with the radius ϵ.

First let us consider the case when S is a segment. Points in B_ϵ that are below segment S have the distance to both points p_1 and p_2 strictly smaller than $1.2a$. This is true because the unit ball R centered in p_1 or p_2 and scaled by the factor $1.2a$ passes trough S. For ϵ small enough all such points will have the distance to p_L and p_R strictly greater than $1.2a$, hence they are in A. This follows from the definition of s.

Now we consider the case when S is a one-point set, i.e. $S = \{s\}$. The unit balls centered in points p_1 and p_2 and scaled by $1.2a$ are overlapping in point s by crossing, not by touching (as in that case there would be another point in S, which yields a contradiction). Hence, there is an area in B_ϵ in which points have the distance to both p_1 and p_2 strictly smaller than $1.2a$. For ϵ small enough all points in that area will have the distance to p_L and p_R strictly greater than $1.2a$, hence they are in A. □

The following lemma is of crucial importance.

Lemma 3.11. *Let the lattice be defined by the fundamental triangle $p_1p_2p_3$. Any lattice point triangle that is not a fundamental triangle has a perimeter in L_R norm greater than $\Delta + 1$.*

Proof. Recall that for the perimeter Δ of the fundamental triangle $3a < \Delta \leq 3.4a$ holds. Furthermore, $a < b, c \leq 1.2a$. Hence if we prove that the L_R distance between any two lattice points, that are not points of the same fundamental triangle, is strictly greater than $\max\{b, c\}$, then every non-fundamental triangle that has perimeter less or equal than Δ, must have two sides of size a. Such a triangle is necessarily a flat triangle defined by three consecutive points on a horizontal line, and has perimeter $4a > \Delta$. Hence, by proving that the distance between any two lattice points, that are not points of the same fundamental triangle, is strictly greater than $\max\{b, c\}$, we get that the perimeter of any non-fundamental triangle is strictly greater than Δ.

Let p be a point in the lattice. For every other point q such that p and q are not points of the same fundamental triangle, we will define a triangle from which, by the triangle inequality, $d_R(p, q) > \max\{b, c\}$ will follow. We divide the lattice into ten areas by the following five lines: an a-line, b-line, c-line, and left and right diagonal line, all of which pass through p. They are represented as solid lines in Figure 3.7, where we w.l.o.g. consider only the upper half plane. Depending in which area point q is positioned, we choose a third point t such that from triangle inequality $d_R(p, q) \geq d_R(p, t) - d_R(q, t) > \max\{b, c\}$ follows.

- q is in the area determined by a-line and left (right) diagonal line (point q_1 (q_5) in Figure 3.7). We choose t to be the point of intersection of border left (right) diagonal and b-line (c-line) that passes through q.
- q is in the area determined by b-line (c-line) and left (right) diagonal line (point q_2 (q_4) in Figure 3.7). We choose t to be the point of

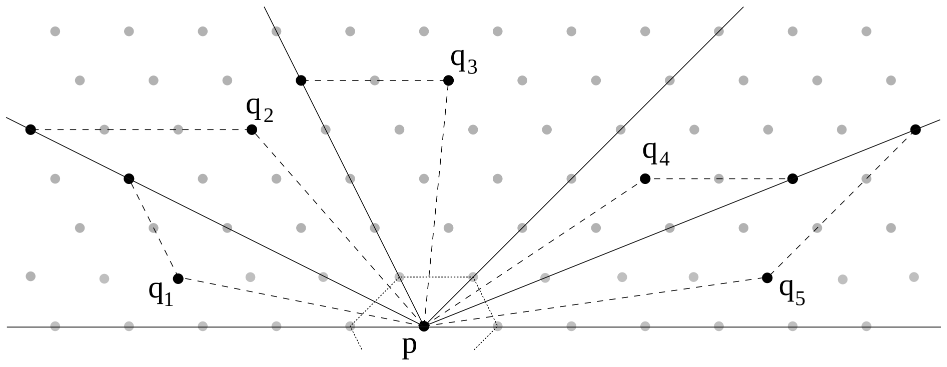

Figure 3.7: Non-fundamental triangles have a large perimeter

intersection of border left (right) diagonal line and a-line that passes through q.

- q is in the area determined by b-line and c-line (point q_3 in Figure 3.7). We choose t to be the point of intersection of a-line that passes through q and border b-line if $b > c$, and border c-line otherwise.

The distance of neighboring lattice points on segment $\overline{pt}$ is strictly greater than the distance of neighboring lattice points on segment $\overline{qt}$. Furthermore, the distance of neighboring lattice points on segment $\overline{pt}$ is greater or equal than $\max\{b, c\}$. Finally, the number of lattice points on segment $\overline{qt}$ is strictly smaller than the number of lattice points on segment $\overline{pt}$, hence $d_R(p, q) \geq d_R(p, t) - d_R(q, t) > \max\{b, c\}$.

If we observe the argument above more carefully, we see that for any two points p and q that are not on the same fundamental triangle, their distance is at least by $\min\{di_L - b, di_R - c, \max\{b, c\} - a\}$ greater than any side size of the fundamental triangle, i.e. $d_R(p, q) - \max\{b, c\} \geq \min\{di_L - b, di_R - c, \max\{b, c\} - a\}$. From the construction of $p_1p_2p_3$ the inequality $\min\{di_L - b, di_R - c, \max\{b, c\} - a\} \geq 1$ holds. Hence it follows that any non-fundamental triangle has perimeter greater than $\Delta + 1$. □

The results of the current section are summarized in the following theorem.

Theorem 3.12. *For any norm L_R with unit ball R in the Cartesian plane $\mathbb{R}^2$, there exist two integer vectors v_1 and v_2, such that the lattice generated by v_1 and v_2 has the following properties.*

(*i*) *The fundamental triangle of the lattice with vertices $p_1 = 0$, $p_2 = v_1$ and $p_3 = v_2$ has a certain perimeter Δ (measured in the L_R norm).*

(*ii*) *Any three (distinct) points q_1, q_2, q_3 in the lattice either form a triangle congruent to the fundamental triangle $p_1p_2p_3$, or otherwise form a triangle with perimeter at least $\Delta + 1$ (measured in the L_R norm).*

3.6 The minimization problem

Throughout this section, we investigate versions of the A3AP with perimeter costs where the elements of $X \cup Y \cup Z$ are points in the Cartesian plane $\mathbb{R}^2$. The distances $d(x, z)$ between points are measured according to some fixed norm L_R with unit ball R.

We will show that for every unit ball R, the corresponding version of the A3AP with perimeter costs is NP-hard. Our reduction is built around the fundamental triangle and the lattice introduced in Theorem 3.12. A *diamond* is a set of four lattice points obtained by gluing together two fundamental triangles along one side; see Figure 3.8. We partition the lattice points into three classes, so that every fundamental triangle contains exactly one point from each class. In the figures the three classes are depicted by circles (○), squares (□) and filled circles (●); see Figure 3.9.

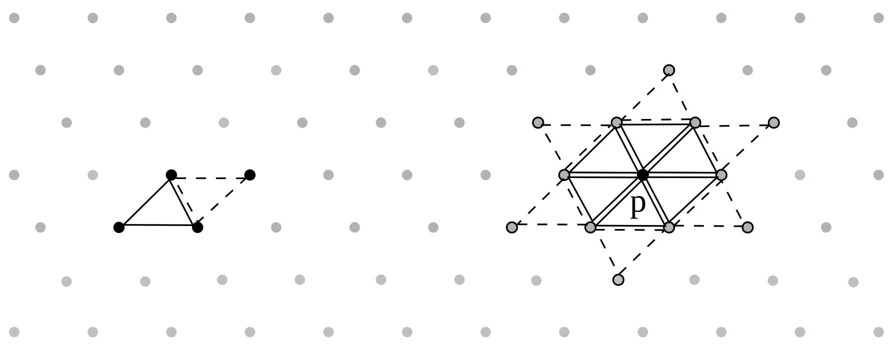

Figure 3.8: A diamond (to the left) and all possible six directions of a diamond incident to point p (to the right)

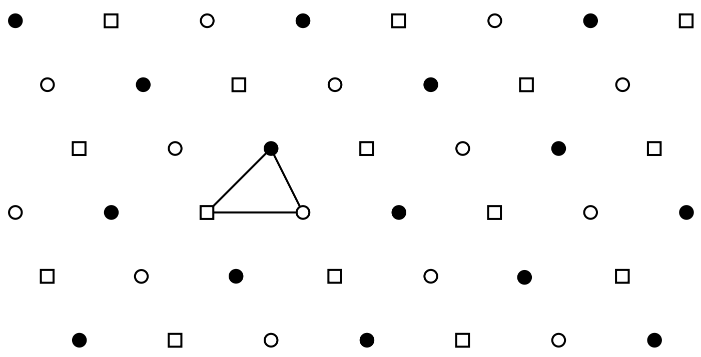
Figure 3.9: The three-colored lattice

Our reduction uses ideas that are similar to those used by Spieksma et al. [77] and Pferschy et al. [62]. The reduction is from the following special case of the A3AP whose NP-hardness has been established by Dyer et al. [34]. To avoid notational collisions between the variables in the A3AP and the variables in the planar-3DM, we will consistently denote objects in the planar-3DM by primed variables. The notion *planar* in the planar-3DM and the P3AP is of a different nature.

> Problem: Planar 3-dimensional matching problem (planar-3DM)
>
> Instance: Three pairwise disjoint sets X', Y' and Z' with $|X'| = |Y'| = |Z'| = q'$ and a set $T' \subseteq X' \times Y' \times Z'$ such that (i) every element of $X' \cup Y' \cup Z'$ occurs in at most three triples from T', and (ii) the corresponding graph G' is planar. (This graph G' contains a vertex for every element of $X' \cup Y' \cup Z'$ and a vertex for every triple in T'. There is an edge connecting a triple to an element if and only if the element is a member of the triple.)
>
> Question: Does there exist a subset T^* of q' triples in T' such that each element of $X' \cup Y' \cup Z'$ is contained in precisely one triple from T^* ?

Hence let us consider an arbitrary instance of the planar-3DM. In the first step, we compute a planar layout of the planar graph G' that maps the vertices of G' into integer points in $\mathbb{Z}^2$ and that maps its edges into straight line segments. This can be done in polynomial time, for instance by using the algorithm of Schnyder [75].

In the second step, we map the planar layout into the three-colored lattice. Every point (α, β) in the planar layout maps into a point that is in the close neighborhood of the point $100\alpha\, v_1 + 100\beta\, v_2$ in the three-colored lattice; here v_1 and v_2 are the integer vectors from Theorem 3.12. Every element of $X' \cup Y' \cup Z'$ is mapped into a corresponding *element point*; every element of X' goes into a circle ($\bigcirc$), every element of Y' goes into a square ($\square$), and every element of Z' goes into a filled circle ($\bullet$). Every triple in T' is mapped into a fundamental triangle called *triple triangle*. These element points and triple triangles roughly imitate the planar layout constructed above; there is plenty of leeway for doing this, since the main restriction is that the various objects should be embedded far away from each other.

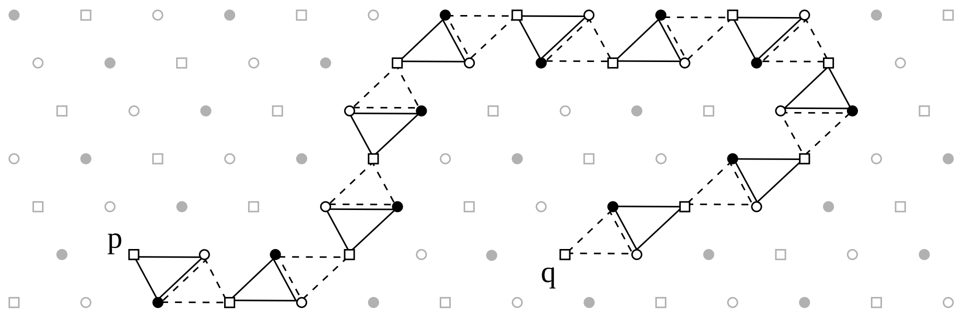

Figure 3.10: A chain of diamonds between two points p and q

In the third step, we introduce several chains of diamonds that connect certain element points to certain triple triangles; see Figure 3.10 for an illustration. Every such chain connects an element point (for some element x' of $X' \cup Y' \cup Z'$) to a triple triangle (whose corresponding triple t' in T' contains that element x'). These chains roughly follow the straight line segment that corresponds to the edge between x' and t' in the planar layout in the first step. Figure 3.11 shows how such a chain is attached to a triple triangle, and Figure 3.12 shows how such a chain is attached to an element point.

Three comments are in place. First, if an element x' of $X' \cup Y' \cup Z'$ occurs in only two triples in T', then the corresponding element point is attached to only two chains of diamonds. Secondly, for every chain of diamonds the attachment point in the triple triangle belongs to the same class ($\bigcirc$, $\square$, $\bullet$)

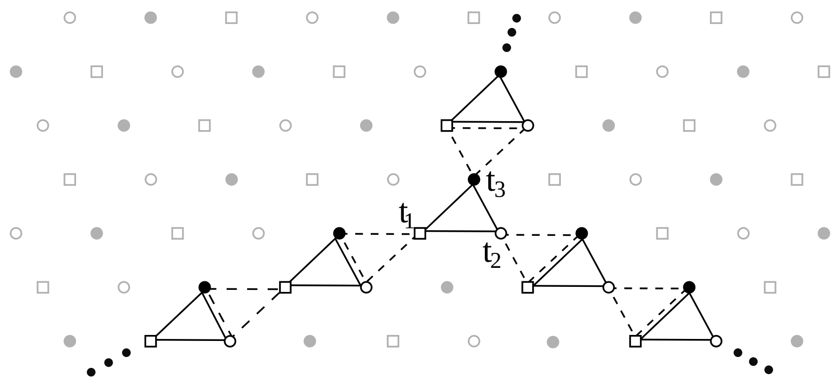

Figure 3.11: How a chain of diamonds attaches to a triple triangle

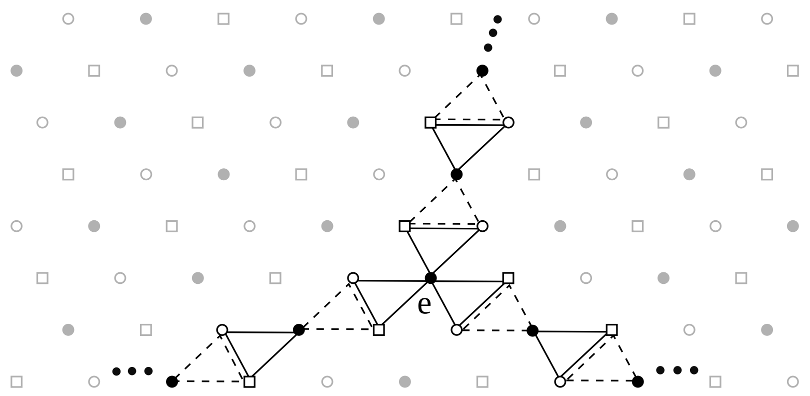

Figure 3.12: How a chain of diamonds attaches to an element point

as the element point at the other end of the chain. Thirdly, we note that there are two combinatorially different ways of choosing a triple triangle in the lattice; one way has the vertices in the classes ○, □, ● clockwise, and the other way has the vertices in the classes ○, □, ● counter-clockwise. We always pick the way that allows a crossing-free attachment of the three chains of diamonds connected to the triple triangle; see Figure 3.13 for an illustration.

The element points, triple triangles and chains of diamonds altogether contain $3n$ points from the three-colored lattice, and each of the three classes contains exactly n points. These three sets with n points form three sets X, Y, Z in an A3AP instance with perimeter costs. We complete the reduction by defining the bound $B = \lceil n\,\Delta \rceil$. The following two lemmas establish the connections between the considered instance of the

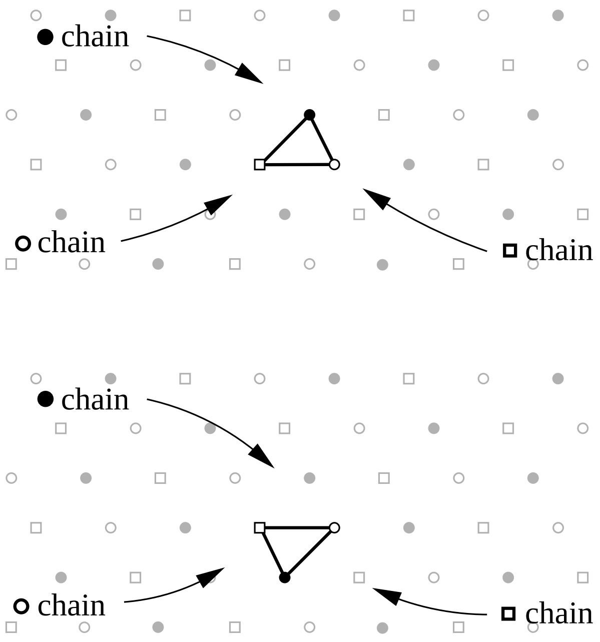

Figure 3.13: Connecting chains to triple triangles: the upper picture shows an infeasible clockwise choice, the picture at the bottom shows the feasible counter-clockwise choice

planar-3DM and the newly constructed instance of the A3AP.

Lemma 3.13. *If the constructed instance of the A3AP has a solution with objective value at most B, then the considered instance of the planar-3DM has answer YES.*

Proof. Assume that the A3AP instance has a solution with objective value at most B. Then by Theorem 3.12 all n triples in this solution have perimeter Δ and form fundamental triangles in the lattice. Moreover, it is straightforward to verify that from any chain of diamonds the solution either picks all the dashed triangles or picks all the solid triangles; see Figures 3.11 and 3.12.

We define a subset T^* of triples in T' by picking all the triples for which the corresponding triple triangle occurs in the solution for the A3AP instance. Consider some element $x' \in X' \cup Y' \cup Z'$. The corresponding element point is contained in exactly one solid triangle in the A3AP solution, and this triangle must belong to some chain; see Figure 3.12. Consequently, this is a chain of solid triangles which propagates to some triple triangle.

Figure 3.11 shows that the corresponding triple triangle is in T^*. To summarize, every element $x' \in X' \cup Y' \cup Z'$ is contained in exactly one triple in T^*. □

Lemma 3.14. *If the considered instance of the planar-3DM has answer YES, then the constructed instance of the A3AP has a solution with objective value at most B.*

Proof. Assume that the planar-3DM instance has answer YES, so that there is a set T^* of q' triples in T' that covers every element of $X' \cup Y' \cup Z'$ exactly once. Then we construct the following solution for the A3AP instance. For every triple in T^*, we pick the corresponding triple triangle for the A3AP solution. For every element $x' \in X' \cup Y' \cup Z'$, we pick the solid triangles in the chain of diamonds that connects the element point for x' to the triple triangle whose triple covers x' in T^*; in the other chains incident to this element point, we pick the dashed triangles. As all points in $X \cup Y \cup Z$ are covered by the n picked (fundamental!) triangles, their overall length equals $n\,\Delta$. □

Note that the bound B in our construction is an integer, and note that all the points in $X \cup Y \cup Z$ have integer coordinates; hence the reduction can be easily implemented in polynomial time (and without worrying about computations with irrational numbers). Together with Lemmas 3.13 and 3.14 this yields the following theorem.

Theorem 3.15. *For any fixed norm L_R with unit ball R in two-dimensional space $\mathbb{R}^2$, the A3AP with perimeter costs measured according to L_R norm is NP-hard.*

Note that the reduction described above allows for the NP-hardness result for the bottleneck-A3AP as well. Indeed, set bound B to be $\lceil \Delta \rceil$. Hence we have the following theorem.

Theorem 3.16. *For any fixed norm L_R with unit ball R in two-dimensional space $\mathbb{R}^2$, the bottleneck-A3AP with perimeter costs measured according to L_R norm is NP-hard.*

3.7 Implications for the weighted 3-dimensional matching problem

Up to this point we have been solely concerned with the axial 3-dimensional *assignment* problem, where the underlying elements belonged to three classes X, Y and Z, and where every triple contained exactly one element from every class. In the closely related weighted 3-dimensional *matching* problem W3DM all the elements belong to the same class: An instance of the W3DM consists of a ground set U with $|U| = 3n$ and a cost function $c : U \times U \times U \to \mathbb{R}$. The goal is to find a set of n triples in $U \times U \times U$ that cover every element in U exactly once, so that the sum of the costs of these triples is minimized. In the maximization version max-W3DM of the W3DM, this sum is to be maximized.

The algorithmic behavior of the W3DM is very similar to that of the A3AP. Both problems are NP-hard in general, and (as a rule of thumb) algorithms for one problem usually translate into algorithms for the other problem. Pferschy et al. [62] proved that the W3DM with perimeter costs under Euclidean distances in $\mathbb{R}^2$ is NP-hard. Our hardness arguments in Section 3.6 can easily be adapted to the W3DM by setting $U := X \cup Y \cup Z$, thus extending and generalizing the result of [62] to arbitrary norms.

Corollary 3.17. *For any fixed norm L_R with unit ball R in two-dimensional space $\mathbb{R}^2$, the W3DM with perimeter costs measured according to L_R norm is NP-hard.*

Corollary 3.18. *For any fixed norm L_R with unit ball R in two-dimensional space $\mathbb{R}^2$, the bottleneck-W3DM with perimeter costs measured according to L_R norm is NP-hard.*

Also the NP-hardness proofs in Section 3.4 for max-A3AP (when the dimension is part of the input) can easily be carried over to the matching problem.

Corollary 3.19. *For any fixed p with $1 \leq p \leq \infty$, the max-W3DM with perimeter costs measured according to the L_p norm is NP-hard.*

In a similar fashion, the positive results in Sections 3.2 and 3.3 for the maximization version carry over to the weighted 3-dimensional matching problem. We leave the (fairly easy) technical details to the reader.

Corollary 3.20. *The max-W3DM with perimeter costs under tunneling distances can be solved within a time complexity that depends polynomially on the instance size n (and exponentially on the number k of tunnels).*

Corollary 3.21. *For any polyhedral norm L_R with unit ball R in s-dimensional space $\mathbb{R}^s$, the max-W3DM with perimeter costs measured according to L_R norm can be solved within a time complexity that depends polynomially on the instance size n (and exponentially on the number k of facets of the polyhedral unit ball).*

Corollary 3.22. *For any (not necessarily polyhedral) norm L_R with unit ball R in s-dimensional space $\mathbb{R}^s$, the max-W3DM with perimeter costs measured according to L_R possesses a PTAS.*

3.8 Conclusions

We have derived a variety of results on the complexity of the A3AP and the max-A3AP with perimeter costs, when distances are measured according to certain norms. The A3AP turned out to be hard for all norms, even if the dimension of the underlying Cartesian space $\mathbb{R}^s$ is $s = 2$. This of course (trivially) implies NP-hardness also for all dimensions $s \geq 3$.

The max-A3AP with perimeter costs shows a more versatile behavior. If the dimension s is fixed, then the max-A3AP is easy for polyhedral norms; furthermore the max-A3AP has a PTAS for all (not necessarily polyhedral) norms. If the dimension s is part of the input, the max-A3AP is NP-hard for any L_p norm.

Open Problem 3.23. Determine whether the max-A3AP with perimeter costs is NP-hard if the elements are points in 2-dimensional space $\mathbb{R}^2$ and if the distances are measured according to the Euclidean norm L_2.

Chapter 4

Special cases of the planar 3-dimensional assignment problem

In this chapter we investigate special cases of the P3AP and its variants. See Section 2.3 for the definition and basic facts about the P3AP. The most important results of this chapter can be found in [30].

4.1 Monge-like structures

The majority of special case structures investigated in this chapter are related to the Monge property. We list various Monge-like structures and point the relations between them.

The most well known Monge structures are Monge matrices which are defined in Definition 2.1. As we are dealing with 3-dimensional assignment problems, 3-dimensional Monge arrays as defined in Definition 2.2 are of special interest to us. Note that a two dimensional Monge array is a Monge matrix. It is well known that a d-dimensional array C is a Monge array if and only if every two-dimensional subarray (matrix) of C corresponding to fixed values for $d-2$ of the d indices in C is a Monge matrix, see [1].

Next we define a special class of Monge arrays.

Definition 4.1. An $n_1 \times n_2 \times \cdots \times n_d$ d-dimensional array $C = (c_{i_1 i_2 \cdots i_d})$

is called *distribution array* if

$$c_{i_1 i_2 \cdots i_d} = -\sum_{j_1=1}^{i_1} \sum_{j_2=1}^{i_2} \cdots \sum_{j_d=1}^{i_d} p_{j_1 j_2 \cdots j_d} \tag{4.1}$$

where $p_{j_1 j_2 \cdots j_d} \geq 0$ for all $j_1, j_2, \ldots, j_d$. Array $P = (p_{i_1 \cdots i_d})$ is called *density array* of C.

It is easy to show that every distribution array is a Monge array. Moreover the following proposition holds, see [22].

Proposition 4.2. *An $n_1 \times n_2$ matrix $M = (m_{ij})$ is a Monge matrix if and only if there exists an $n_1 \times n_2$ distribution matrix (i.e. two dimensional distribution array) $D = (d_{ij})$ and two vectors $U = (u_i)$ and $V = (v_i)$ such that*

$$m_{ij} = d_{ij} + u_i + v_j \qquad \text{for all } i, j.$$

There is no relation between Monge and distribution arrays analogous to Proposition 4.2 in the case when the dimension is greater than two.

Next we consider a superclass of the class of 3.dimensional Monge arrays.

Definition 4.3. Let $C = (c_{ijk})$ be a three-dimensional $n \times n \times p$ array such that for every k, $1 \leq k \leq p$, the matrix $M^k = (m_{ij}^k)$ where $m_{ij}^k = c_{ijk}$ is a Monge matrix. Then C is called an *array with Monge layers*, or *layered Monge array* for short. If M^k is a bottleneck Monge matrix for all k, then C is called *layered bottleneck Monge* array.

$$\begin{array}{ccccc} \text{distribution} & & \text{Monge} & & \text{layered} \\ & \subset & & \subset & \\ \text{3-dim. arrays} & & \text{3-dim. arrays} & & \text{Monge arrays} \end{array}$$

Figure 4.1: Relation between array classes

4.2 Intractability results on Monge-like arrays

Monge arrays make the axial 3-dimensional assignment problem (A3AP) easily solvable, see Proposition 2.3. For the planar 3-dimensional assignment problem (P3AP) this is not the case. Before we state the intractability results, we present the following straightforward observation which will be useful later.

Observation 4.4. Let C be an $n \times n \times p$ array and let $A = (a_{ij})$ be an $n \times n$ matrix and $B = (b_{ij})$ and $D = (d_{ij})$ be $n \times p$ matrices.

- Let $p = n$. The P3AP instances with cost array C and with cost array $C + C'$ where

  $$c'_{ijk} = a_{ij} + b_{ik} + d_{jk} \qquad \text{for all} \quad i, j, k \in \{1, \dots, n\} \tag{4.2}$$

  are equivalent. The objective function value is shifted by the constant $\alpha = \sum_{i=1}^{n} \sum_{j=1}^{n} (a_{ij} + b_{ij} + d_{ij})$.

- The p-P3AP instances with cost array C and with cost array $C + C''$ where

  $$c''_{ijk} = b_{ik} + d_{jk} \qquad \text{for all} \quad i, j \in \{1, \dots, n\}, k \in \{1, \dots, p\} \tag{4.3}$$

  are equivalent. The objective function value is shifted by the constant $\beta = \sum_{i=1}^{n} \sum_{k=1}^{p} (b_{ik} + d_{ik})$.

In Subsection 5.2.1 arrays that fulfill the property (4.2) are named sum-decomposable with parameters $d = 3$ and $s = 2$. There it is proved that all feasible solutions of the P3AP have the same cost if and only if the cost array C is sum-decomposable with parameters $d = 3$ and $s = 2$. The cost arrays that fulfill (4.3) play an analogous role for the p-P3AP for the $p < n$ case.

Theorem 4.5. *The P3AP stays NP-hard on the class of layered Monge cost arrays.*

Proof. Consider the $n \times n$ matrix $M = (m_{ij})$ with $m_{ij} = -(i + j)^2$ (or alternatively $m_{ij} = 4n^2 - (i + j)^2$ if one prefers to deal with nonnegative cost arrays). It is easy to check that M is a Monge matrix. Let $F = (f_{ijk})$ be the $n \times n \times n$ cost array obtained from M by setting $f_{ijk} = m_{ij}$ for all $i, j, k \in \{1, \dots, n\}$. Since M is a Monge matrix, F is a layered Monge array.

Let C be an arbitrary $n \times n \times n$ 0-1 array. Consider the $n \times n \times n$ cost array $C' = (c'_{ijk})$ with $c'_{ijk} = f_{ijk} + c_{ijk}$. It is easy to check that the layered Monge array property is inherited to C' from F.

It follows from Observation 4.4 that the P3AP on cost matrix C is equivalent to the P3AP on cost array C'. Since the P3AP is NP-hard for general 0-1 cost arrays it follows that the P3AP stays hard when restricted to layered Monge arrays. □

Theorem 4.6. *The p-P3AP stays NP-hard on the class of layered Monge cost arrays.*

Proof. The proof builds on the idea used in the proof of Theorem 4.5. We make again use of the $n \times n$ Monge matrix $M = (m_{ij})$ with $m_{ij} = -(i+j)^2$. Let $n' = 2n$ and expand M into an $n' \times n'$ matrix M' as follows:

$$M' = \begin{pmatrix} M & Y \\ Y^t & Z \end{pmatrix}$$

where Z is the $n \times n$ zero matrix and $Y = (y_{ij})$ is the $n \times n$ matrix with $y_{ij} = i \cdot n$ for all $i, j \in \{1, \dots, n\}$. It is easy to check that M' is again a Monge matrix.

Let an instance of the P3AP with an $n \times n \times n$ 0-1 cost array C be given. Expand C into an $n' \times n' \times n$ array $\widehat{C} = (\widehat{c}_{ijk})$ by defining

$$\widehat{c}_{ijk} = \begin{cases} c_{ijk} & \text{for } i, j, k \in \{1, \dots, n\} \\ 0 & \text{for } i, j \in \{n+1, \dots, n'\}, k \in \{1, \dots, n\}. \end{cases}$$

Now create the $n' \times n' \times n$ cost arrays $F' = (f'_{ijk})$ and $C' = (c'_{ijk})$ with

$$f'_{ijk} = m'_{ij} \quad \text{for all } i, j \in \{1, \dots, n'\}, k \in \{1, \dots, n\}$$

and

$$c'_{ijk} = f_{ijk} + \widehat{c}_{ijk} \quad \text{for all } i, j \in \{1, \dots, n'\}, k \in \{1, \dots, n\}.$$

It is easy to check that C' is a layered Monge array. Now consider the instance I of the p-P3AP with cost array C' array and $p = n$.

Note that the cost entries in the Y and the Y^t block of M' (which carry over to the k-planes of F' and C') are positive numbers $\geq n$ while the entries in Z are zero and the entries in M are negative. Thus an optimal solution of the p-P3AP will contain only elements (i, j, k) for which either $i, j \in \{1, \ldots, n\}$ or $i, j \in \{n + 1, \ldots, n'\}$ holds when such solutions exist (which is easily seen to be the case). Hence an optimal solution of the P3AP with cost array C can be obtained from an optimal solution of the p-P3AP instance I by dropping all 3-tuples in the solution that involve indices $> n$. The result now follows from the NP-hardness of the P3AP for 0-1 cost arrays. $\square$

Note that the cost arrays used in the hardness reductions in the proofs of Theorems 4.5 and 4.6 do not belong to the class of Monge arrays.

Open Problem 4.7. Determine the complexity of the P3AP and the p-P3AP on the class of Monge arrays and its subclass of distribution arrays.

4.3 The optimal solution structure of the p-P3AP on layered Monge arrays

Note that in the hard instance constructed in the proof of Theorem 4.6 the number of layers p is of the order of n. This leaves the complexity status of the p-P3AP for layered Monge arrays unsettled when p is a constant. In this section we present results on the structure of the optimal solution of the p-P3AP on layered Monge arrays. The bandwidth result shown in Theorem 4.12 will be used in Section 4.4 to provide an algorithm that solves the p-P3AP on layered Monge arrays in polynomial time if p is fixed.

4.3.1 Block structure result for the 2-P3AP

In this subsection we will prove that there always exists an optimal solution of the 2-P3AP on layered Monge arrays which fulfills a regular block structure. To formulate this result and to prove it we will need the following definitions. It will be convenient to represent feasible solutions of the 2-P3AP by Latin rectangles, cf. Subsection 2.3.1.

Definition 4.8. A set B of the j-th to the m-th column of a Latin rectangle L is called a *block* if the following is satisfied. The columns in B contain only integers from j to m, first $j-1$ columns of L contain only integers from 1 to $j-1$, and B is minimal with respect to the previous two properties.

Note that for every Latin rectangle L, the set of all its blocks forms a partition of L. Figure 4.2 illustrates an example with a 2×12 Latin rectangle. It is partitioned into 4 blocks.

1	2	4	3	5	8	7	9	6	11	10	12
2	1	3	5	4	9	6	7	8	10	12	11

Figure 4.2: Blocks of a Latin rectangle

Note that if a block B consists of m columns $j, \ldots, j+m-1$ then the following property is fulfilled for all $i = 1, \ldots, m-1$:

($*$) The first i columns of B contain an integer $x > i + j$ and the last i columns of B contain an integer $x < j + m - i$.

The local operation of a swap that exchanges two integers in a row of an $p \times n$ Latin rectangle will play a fundamental role in what follows.

Definition 4.9. Let X be a feasible solution of the p-P3AP and let $r, q \in \{1, \ldots, n\}$, $r < q$. The operation that exchanges the positions of r and q in a row k of the Latin rectangle representation of X is referred to as a *swap* and is denoted by $\mathrm{SWAP}(r, q, k)$. The swap is called *feasible* if the newly obtained solution X' is feasible. The swap is called *non-increasing* if $c(X') \leq c(X)$.

Lemma 4.10. *Let C be an $n \times n \times p$ layered Monge array and r, q be integers such that $1 \leq r < q$. Then the swap $\mathrm{SWAP}(r, q, k)$ is non-increasing if r is placed to the right of q.*

Proof. Let q be placed in column j and row k and $r < q$ be placed in column $s > j$, i.e. $x_{qjk} = 1$ and $x_{rsk} = 1$. Since C is a layered Monge array, the following property is fulfilled: $c_{rjk} + c_{qsk} \leq c_{rsk} + c_{qjk}$. It follows that exchanging the position of r and q in row k will lead to a new (not necessarily feasible) solution X' with $x'_{rjk} = 1$ and $x'_{qsk} = 1$ such that $c(X') \leq c(X)$. □

$$r < q \quad \Longrightarrow \quad \begin{array}{|c|c|c|c|c|c|}\hline & r & & q & & \\ \hline & & & & & \\ \hline\end{array} \leq \begin{array}{|c|c|c|c|c|c|}\hline & q & & r & & \\ \hline & & & & & \\ \hline\end{array}$$

Figure 4.3: Non-increasing SWAP

Theorem 4.11. *For the* 2*-P3AP with an* $n \times n \times 2$ *cost array* C *which is a layered Monge array, there always exists an optimal solution which represented as Latin rectangle decomposes into blocks with* 2 *or* 3 *columns.*

Proof. For $n \leq 3$ there is nothing to show. For $n \geq 4$ we will show that every feasible solution X can be transformed to a feasible solution X' which has the claimed property and for which $c(X') \leq c(X)$ holds.

Our block transformation approach works iteratively. We decompose blocks into smaller blocks step by step.

First, we will show that a block B of size 4 can be transformed into two blocks of size 2 by performing a series of non-increasing swaps. Suppose without loss of generality that the smallest integer in the block B is 1 (hence the remaining integers in B are $2, \dots, n$, where n is the size of the block B). Depending on the number of occurrences of 1 in the first two columns, we distinguish the following three cases **(1)**, **(2)** and **(3)**.

(1): There are two occurrences of 1 in the first two columns.

W.l.o.g. will assume that 1's are on the main diagonal, cf. table (I).

$$\text{(I)} \quad \begin{array}{|c|c|c|c|}\hline 1 & a & & \\ \hline b & 1 & & \\ \hline\end{array} \qquad\qquad \text{(II)} \quad \begin{array}{|c|c|c|c|}\hline 1 & a & c & b \\ \hline b & 1 & c & a \\ \hline\end{array}$$

If $a = b$, the block B splits up into two 2×2 blocks and we are done. So assume that $a \neq b$. Now we search for the second occurrences of a and b in block B. Note that a and b cannot appear in the same column because then the solution would be infeasible (cf. table (II)). If $a > b$ we perform SWAP$(b, a, 1)$ and if $a < b$ we perform SWAP$(a, b, 2)$, which are both non-increasing swaps. Table (III) illustrates the first case.

$$\text{(III)} \quad \begin{array}{|c|c|c|c|}\hline 1 & \mathbf{a} & c & \mathbf{b} \\ \hline b & 1 & a & c \\ \hline\end{array} \quad \longrightarrow \quad \begin{array}{|c|c|c|c|}\hline 1 & \mathbf{b} & c & \mathbf{a} \\ \hline b & 1 & a & c \\ \hline\end{array}$$

If c is smaller than a and b, then using two swaps exchange the positions of c and $\min\{a, b\}$.

(2): There is only one occurrence of 1 in the first two columns.

It does not matter in which row 1 appears. We consider the two subcases that arise depending on whether 1 occurs in the first or in the second column.

(2.1): 1 occurs in the first column of B.

1	a		
b	c		

We will divide this case into three subcases **(2.1.1)**, **(2.1.2)** and **(2.1.3)**.

(IV)

1	a		
a	c		1

(V)

1	a	c	b
b	c	1	a

(VI)

1	a	b	c
b	c	a	1

(2.1.1): In this case $a = b$ holds. The column in which 1 occurs for the second time is not containing c because then the last leftover column would contain the same integer in both rows, cf. table (IV). Therefore we can perform SWAP$(1, c, 2)$ and move to the situation from case **(1)**.

(2.1.2): $a \neq b$ and the second 1 appears in the third column of B. If the third column does not contain c, then we perform SWAP$(1, c, 2)$ and again end up with case **(1)**. Otherwise we have a situation like in table (V). In that case first we perform SWAP$(b, c, 1)$ followed by SWAP$(1, c, 2)$ if $b < c$, and SWAP$(c, b, 2)$ followed by SWAP$(1, b, 2)$ if $b > c$. Again we end up with case **(1)**.

(2.1.3): $a \neq b$, and the second 1 appears in the last column. If the last column does not contain c, then we perform SWAP$(1, c, 2)$ and end up with case **(1)**. Otherwise we have a situation like in the table (VI). In that case we perform SWAP$(1, a, 2)$ followed by SWAP$(1, c, 2)$. Again we end up with case **(1)**.

(2.2): 1 appears in the second column.

a	1		
c	b		

We again distinguish two cases.

(2.2.1): If $a = b$, the same argument as in **(2.1.1)** can be applied.

(2.2.2): If $a \neq b$, then we perform SWAP$(1, a, 1)$ and end up with case **(2.1)**.

(3): 1 does not occur in columns 1 and 2.

a	b		1
			c

In this case we just preform SWAP$(1, a, 1)$ or SWAP$(1, b, 1)$, depending on which one preserves feasibility (such swap always exists), and then go to **(2).**

Next we show how to transform any block B of size > 4 into smaller blocks such that at the left end a small block of size 2, 3 or 4 appears. The remaining part (right tail) is then decomposed further in an iterative manner. If all blocks are of size 2 or 3 the procedure stops. As for blocks of size 4, the transformation performs a series of non-increasing swaps.

We distinguish the following three cases **(1')**, **(2')** and **(3')**.

(1'): There are two occurrences of 1 in the first two columns. Again for both configurations of 1's the same arguments can be used because of symmetry. So let us assume that we have the situation from table (VII).

(VII)

1	a	
b	1	

(VIII)

1	a	b
b	1	a

If $a = b$, then by performing SWAP$(2, a, 1)$ and SWAP$(2, a, 2)$ the first two columns will form a block of size 2, and we are left with $n-2$ columns to which we can apply our approach iteratively. Assume that

$a \neq b$. If the second occurrences of a and b do not appear in the same column, we can move $\min\{a, b\}$ to the first or second column by an appropriate swaps and are finished as above.

Hence, it remains to deal with the case shown in table (VIII). We will divide this case into three subcases **(1.1')**, **(1.2')** and **(1.3')**:

(1.1'): The second occurrences of a and b are in the third column, cf. table (IX). The first three columns contain only 3 integers, hence by SWAP$(2, a, i)$ and SWAP$(3, b, i)$, for $i = 1, 2$, we get a 2×3 block and we are done.

(IX)

1	a	b	
b	1	a	

(X)

1	a	2	b	
b	1		a	2

(1.2'): The second occurrences of a and b are not in the third column, and a and b are not 2, cf. table (X). In this case we just perform SWAP$(2, a, 1)$ and SWAP$(2, b, 2)$.

(1.3'): The second occurrences of a and b are not in the third column, and either a or b is 2, cf. table (XI). In this case swap the second occurrence of 2 with any integer in front of it that is not in the first two columns. Then we can perform a swap that moves it into first two columns.

(XI)

1	2		b	
b	1		2	

(XII)

1	a	c
b	c	1

(2'): 1 appears once in the first two columns.

We can assume that 1 appears in the first row and first column. A symmetric argument applies to the other cases. If the other occurrence of 1 and c is not in the same column (unlike in XII), then we can perform SWAP$(1, c, 2)$ and go to **(1')**.

(2.1'): Second occurrence of 1 and c are in the same column, and that is not the third column, cf. table (XII). Then we swap 1 with an integer to the left of it and then move it to the second column, and go to **(1')**.

(2.2)': 1 and c are in the third column, cf. table (XIII). We will cover all the possibilities in 5 cases.

(XIII)

1	a	c	
b	c	1	

(XIV)

1	a	c	b
b	c	1	a

(2.2.1'): If $a = b$, then by performing SWAP$(2, a, i)$ and SWAP$(3, c,$ $i)$, for $i = 1, 2$, the first three columns form a block.
In the remaining four cases we have $a \neq b$.

(2.2.2'): Second occurrences of a and b are not in the same column, cf. table (XIV). Perform SWAP$(b, a, 1)$ if $b < a$ and SWAP$(a, b, 2)$ otherwise, and go to **(2.2.1')**.
In the remaining cases second occurrences of a and b are in the same column.

(2.2.3'): Second occurrences of a and b are in the forth column. By performing SWAP$(2, a, i)$, SWAP$(3, b, i)$ and SWAP$(4, c, i)$, for $i = 1, 2$, the first four columns form a block, and we have shown in the first part of the proof how to decompose blocks of size four.

(XV)

1	a	c	2 b
b	c	1	a 2

(XVI)

1	a	c	2
2	c	1	a

(2.2.4'): Second occurrence of a and b is not in the fourth column, and both a and b are not 2 (or 3 if $c = 2$), cf. table (XV). In this case we perform SWAP$(2, a, 1)$, SWAP$(2, b, 2)$ and SWAP$(3, c, i)$, for $i = 1, 2$, and we are finished.

(2.2.5'): a or b is 2 (or 3 if $c = 2$), cf. table (XVI). Now we swap the second occurrence of 2 with the integer to the left of it, and then perform SWAP$(2, a, 1)$ and we are finished.

(3'): 1 does not appear in the first two columns.

Just swap one 1 with an integer from the first or the second column, depending on which swap preserves feasibility, and go to **(2')**.

We covered all the cases, which concludes the proof. □

For the special case of distribution arrays we have a more elegant proof which is based on the fact that for distribution arrays there exists an explicit formula for the cost of a feasible solution in terms of entries of the density array. As Theorem 4.11 is more general, we omit the proof for the special case.

Theorem 4.11 can be used to design a polynomial time algorithm, see Section 4.4.

4.3.2 Example with a single large block for $p \geq 3$

It is natural to ask whether the block structure result for the 2-P3AP from Theorem 4.11 carries over to the 3-P3AP. This is not the case, not even for the subclass of distribution arrays as is demonstrated by the following counterexample in which the unique optimal solution consists of a single large block. Note that the block structure of the Latin rectangle representation corresponds to the block diagonal structure of the partial Latin square representation.

Let $n = 10$ and $p = 3$. We consider the distribution array C which is generated by the density array $P = (p_{ijk})$ defined by

$$p_{ij1} = \begin{cases} 100 & \text{if } i = j = 7 \\ 1 & \text{otherwise} \end{cases}, \quad p_{ij2} = \begin{cases} 10a & \text{if } i = 4 \text{ and } j = 5 \\ 10a^3 & \text{if } i = 9 \text{ and } j = 10 \\ b_j & \text{otherwise} \end{cases},$$

and $p_{ij3} = a^a$ for all $i, j \in \{1, \dots, 10\}$, where $B = (b_i) = (1, 1, a, a, a, a^2, a^2, a^3, a^3, a^3)$ and a is some large number. For notational convenience let us refer to the 10×10 matrix that corresponds to the k-plane of P as Q_k (i.e.

$q^k_{ij} = p_{ijk}$). We claim that the following 3×10 Latin rectangle L:

$$L = \begin{array}{|cccccccccc|} \hline 3 & 4 & 1 & 2 & 6 & 5 & 8 & 10 & 7 & 9 \\ \hline 2 & 1 & 4 & 5 & 3 & 7 & 6 & 9 & 10 & 8 \\ \hline 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \hline \end{array} \tag{4.4}$$

is the unique optimal solution for the 3-P3AP instance with cost array C. This can be checked by computational means. Below we provide the rationale behind why L shows up as unique optimal solution.

Consider the $10 \times 10 \times 2$ subarray C_2 of C where the k-plane of C with $k = 1$ is dropped. We claim that the following Latin rectangle is an optimal solution of the 2-P3AP on C_2:

$$\begin{array}{|cc|ccc|cc|ccc|} \hline 2 & 1 & 4 & 5 & 3 & 7 & 6 & 9 & 10 & 8 \\ \hline 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ \hline \end{array}. \tag{4.5}$$

This is easy to check. Note that the entries of Q_3 are much larger than the entries in Q_2 and Q_1. It can easily be checked that it thus pays off to choose the overall best permutation for any Monge matrix, namely the identity permutation, as solution for the plane Q_3. By choosing any other permutation more would be lost than can be won for the planes Q_1 and Q_2. Hence the second row of (4.5) has to be the identity permutation.

Because of the structure of the vector B and its role in Q_3, Latin rectangle (4.5) splits into blocks of sizes 2, 3, 2 and 3, respectively. The two entries of Q_2 which involve the multiplicative factor 10 determine the type of the two blocks of size 3 in (4.5).

Since Q_1 has much smaller entries than Q_2 and Q_3, the optimal solution of the $10 \times 10 \times 3$ P3AP with cost array C will have (4.5) as last two rows. This is true since every $r \times n$ Latin rectangle can be extended to an $n \times n$ Latin square [46]. Because $p_{1,7,7} = 100$, the rightmost four columns of the first row in the optimal Latin rectangle L have to be filled with integers that are greater or equal than 7, if possible. There indeed exist three ways to achieve this goal and we take the one with the lowest cost, i.e. the block (8,10,7,9), and fill the rest of the first row of the Latin rectangle in an obviously unique optimal manner. Hence the Latin rectangle L is the unique optimal solution.

As the Latin rectangle L consists of a single block of size $n = 10$, this destroys any hope for a result along the lines of Theorem 4.11 for $p \geq 3$. The same approach can be used to construct examples where an arbitrarily large block arises in the optimal solution. Just add an arbitrary number of 2×2 blocks to the middle of the candidate solution (4.5).

The construction above can be carried over for $p > 3$. Add more k-planes on top of P with much smaller values than in the first three k-planes. Then the optimal solution will have a Latin rectangle representation such that the first three rows are as in the example provided for $p = 3$, use Hall's theorem [46].

4.3.3 Bandwidth result for the p-P3AP

Recall that for the linear assignment problem on a Monge matrix the identity permutation provides an optimal solution, cf. Proposition 2.3. A feasible solution of the p-P3AP is a set of p pairwise disjoint permutations (assignments). Hence, one might expect that for the p-P3AP on a layered Monge array the filled cells of the partial Latin square representation of an optimal solution tend to group around the main diagonal. It is shown below that this is indeed the case.

Theorem 4.12. *Let I be an instance of the p-P3AP with the $n \times n \times p$ layered Monge cost array C. There exists an optimal solution X for I such that the partial Latin square L that represents X has the following property:*

(P) *Whenever the cell in the i-th row and the j-th column of L ($L(i,j)$) is filled, we have that $|i - j| \leq 2p - 2$.*

Proof. Consider an optimal solution X and its representation as partial Latin square L. We start at the upper right corner and shift a line parallel to the main diagonal towards the center of the partial Latin square until one hits for the first time a filled cell. Choose such cell and call it *pivotal cell*. Suppose that the pivotal cell is $L(i,j)$ and contains integer k, see Figure 4.4 for an illustration. We define four sub-rectangles of the partial Latin square L as follows, depending in which of the four quadrants around $L(i,j)$ the cells are located.

- Q_{la}: Contains all cells (r,q) with $r < i$ and $q < j$.

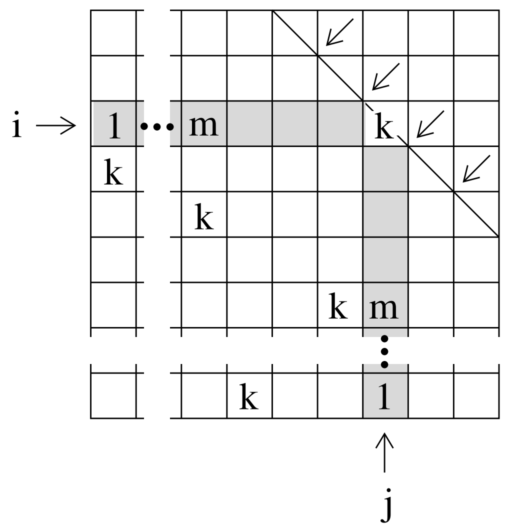

Figure 4.4: Pushing elements closer to the diagonal

- Q_{lb}: Contains all cells (r, q) with $r > i$ and $q < j$.
- Q_{ra}: Contains all cells (r, q) with $r < i$ and $q > j$.
- Q_{rb}: Contains all cells (r, q) with $r > i$ and $q > j$.

Let us refer to Q_{lb} as *candidate area.* If there exists a cell $L(q, r)$ in the candidate area that is filled with integer k while the cells $L(i, r)$ and $L(q, j)$ are empty, we can fill the cells $L(i, r)$ and $L(q, j)$ with integer k and restore the cells $L(i, j)$ and $L(q, r)$ to the empty state. The new solution X' is feasible again and since the cost array C is a layered Monge, it follows that

$$c_{irk} + c_{qjk} \leq c_{ijk} + c_{qrk}.$$

Hence the cost does not increase if we move to the new solution.

Next we examine for which i and j it is always possible to perform a move of the type described above. First we count how many cells in the candidate area are filled with k. Since there are no filled cells in Q_{ra} by assumption, it follows that k occurs in Q_{la} exactly $i-1$ times. Analogously, it follows that k occurs in Q_{rb} exactly $n - j$ times. Consequently, k occurs in the candidate area exactly $n - (i - 1) - (n - j) - 1 = j - i$ times.

Note that filled cells in the part of row i left of the pivotal cell and filled cells in the part of column j below the pivotal element (see the gray area in

Figure 4.4) can lead to the situation in which a candidate integer k cannot be used to perform a move of the type described above. Note that there are exactly $p-1$ filled cells in row i left of $L(i,j)$ and exactly $p-1$ filled cells in column j below $L(i,j)$. Therefore, if

$$j - i > 2p - 2$$

holds, we can always choose an integer such that the performed move is feasible.

Similarly, if we consider the elements below the diagonal, we get that a feasible move can be performed if $i - j > 2p - 2$. Therefore, there always exists an optimal solution such that for all filled cells $L(i,j)$ we have

$$|i - j| \leq 2p - 2.$$

□

In Section 4.4, Theorem 4.12 is used to design a dynamic programming algorithm that solves the p-P3AP on layered Monge arrays in linear time for fixed p.

Note that if the partial Latin square L has property (P) from Theorem 4.12, then L is a band matrix with bandwidth $\leq 4p-3$. Figure 4.5 depicts a feasible solution of the 3-P3AP of size $n = 16$ which satisfies property (P). It is natural to ask whether this bound is tight. For $p = 2$ and odd n it is easy to see that this is the case. It suffices to choose the cost array C such that in the optimal partial Latin square all diagonal entries will be filled by 1's, which is easy to achieve.

We conjecture that the bound $4p-3$ is tight for infinitely many values of p. Moreover, we have constructed a class of instances for which we conjecture that the bound is achieved when p is prime. We have checked the correctness of the conjecture for values of p up to 31 by computational means. Work on the proof is in progress. Now we present this construction.

Let F be an arbitrary feasible solution represented as partial Latin square. First we describe how to construct a layered Monge array C for which F provides an optimal solution to the p-P3AP. Each of the p layers of C will be a distribution matrix. We obtain C by providing the density matrices

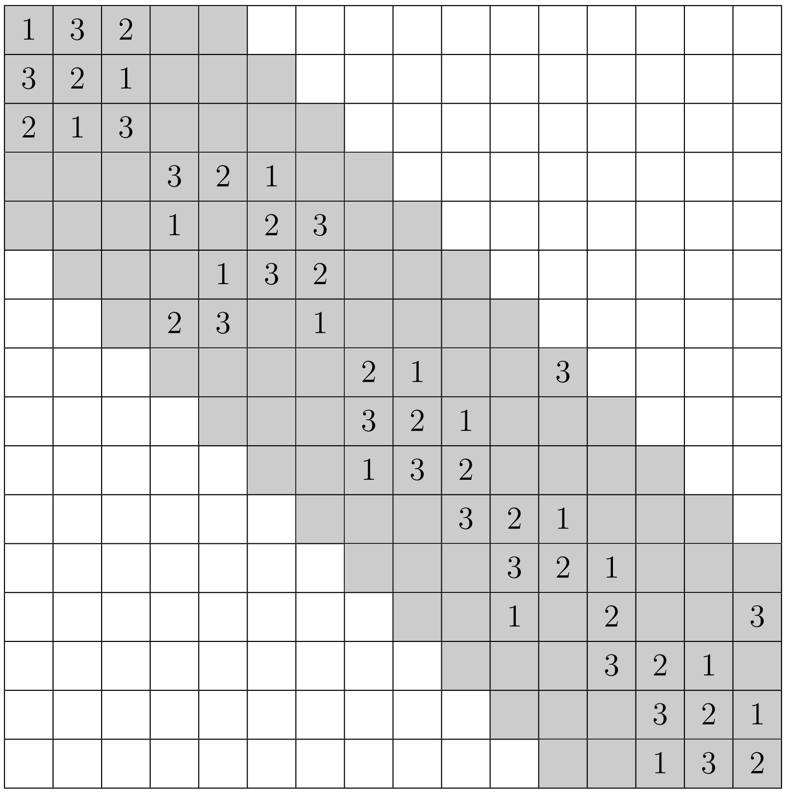

Figure 4.5: A feasible solution of the 3-P3AP

for these p distribution matrices. For each $k \in \{1, \dots, p\}$ let $A^k = (a^k_{ij})$ denote the density matrix for the k-th layer, i.e. the density matrix of the distribution matrix defined by the elements c_{ijk}, $i, j = 1, \dots, n$. Now observe the positions of the occurrences of k in F. If some integer k is on the position (i, j), then it corresponds to the cost element c_{ijk} that satisfies $c_{ijk} = \sum_{r=1}^{i} \sum_{s=1}^{j} -a^k_{rs}$. Hence it corresponds to ij elements from A^k. Let $O^k = (o^k_{ij})$ be a matrix that counts all such occurrences of elements a^k_{ij} for all integers k in F, i.e. o^k_{ij} is the number of k's in F that are on positions (r, s) for which $r \geq i$ and $s \geq j$. Finally, let $ID = (d_{ij})$ be an analogue of the O^k matrix in the case when all k's are on the main diagonal. Now we set a^k_{ij} to 1 if $o^k_{ij} = d_{ij}$, and to 0 otherwise. If we do this for all $k = 1, \dots, p$, then F is an optimal solution for the array C, and we say that C is the *maximal layered Monge array for F.*

Next we describe the partial Latin squares to which we apply the construction above. Consider the Latin square of size p with the first row being $1, 2, \dots, p$, and the i-th row being the first row shifted to the right by $i - 1$.

Take three copies of such Latin square to be diagonal blocks of a partial Latin square of the size $n = 3p$. Further on, we make the following swaps. Move p's from positions $(1, p)$, $(p + 1, 2p)$, $(2p, 2p - 1)$ and $(2p + 1, 3p)$ to positions $(1, 2p - 1)$, $(p + 1, p)$, $(2p, 3p)$ and $(2p + 1, 2p)$. We denote the resulting partial Latin square as the *target solution*, see Figure 4.6. Note

1	2	3	4					5						
5	1	2	3	4										
4	5	1	2	3										
3	4	5	1	2										
2	3	4	5	1										
				5	1	2	3	4						
					5	1	2	3	4					
					4	5	1	2	3					
					3	4	5	1	2					
					2	3	4		1					5
									5	1	2	3	4	
										5	1	2	3	4
										4	5	1	2	3
										3	4	5	1	2
										2	3	4	5	1

Figure 4.6: Target solution for $p = 5$

that if the target solution is the only optimal solution for some cost array, then the bound of Theorem 4.12 is achieved. Indeed, p in the first row achieves the bound.

Conjecture 4.13. *Consider the p-P3AP on the maximal layered Monge array for the target solution. Then the target solution is the only optimal solution if and only if p is prime.*

The necessity can be easily shown. Namely, let p be divisible by $a \in \{2, 3, \ldots, p - 1\}$. Starting from a target solution one can obtain a new optimal solution that does not achieve the bound by applying the following p/a integer rearrangements. The rearrangements only include integers in the middle *block* on the diagonal and the integer p in the first row. In every such rearrangement we take two same value integers on positions (i, j) and (r, s), and move them to positions (i, s) and (r, j). After each of the first

$p/a - 1$ steps there will be a cell containing two integers, but the last step will restore the feasibility. In the first step move integer p in the first row and integer p in the a-th row of the middle block. Then move integers $p - a$ in the a-th and the $2a$-th row of the middle block. Then move the integers $p - 2a$ in rows $2a$ and $3a$, and so on. Due to the Monge property all the steps will preserve the optimality, and the last step will restore the feasibility.

Computer testings have shown that for prime p up to 31 the conjecture holds.

Cost arrays described above are of size $n = 3p$, but they are easily extended to all $n \geq 4p$.

Note that the construction above, i.e. the construction of a layered Monge cost array that fulfills the bound of Theorem 4.12 can be analogously done for distribution arrays as well. Just let $D = (d_{ijk})$ be the density array such that $d_{ijk} = a_{ij}^{k}\alpha^{k}$ for some large constant α.

4.4 Algorithms for the p-P3AP on layered Monge arrays

Using Theorem 4.11 we can design a simple dynamic programming algorithm that solves the 2-P3AP on layered Monge arrays in linear time. Algorithm 4.1 which is described below is one such algorithm.

It easy to see that the complexity of Algorithm 4.1 is $O(n)$ as the number of blocks of size 2 and 3 for a given two and three integers, respectively, is a constant. If the cost array is a Monge array or a distribution array then not all blocks need to be considered, which can be used to speed-up the algorithm.

Although for $p \geq 3$ there does not need to exist an optimal solution with a block structure, based on Theorem 4.12 we can devise a dynamic programming algorithm that finds an optimal solution of the p-P3AP on layered Monge arrays. The running time of the algorithm will be linear in n and exponential in p.

To be able to formulate the algorithm more simply, we express Theorem 4.12 in terms of Latin rectangle representations.

Corollary 4.14. *Let I be an instance of the p-P3AP with the $n \times n \times p$ layered Monge cost array C. There exists an optimal solution X for I such that the Latin rectangle L that represents X has the following property:*

(*P'*) *For $i = 1, \dots, n$, integers i appear only in $(i-2p+2)$-th to $(i+2p-2)$-th column of L.*

Algorithm 4.1: 2-P3AP on layered Monge arrays

```
**value**[]: $value[j]$ stores the optimal value of a feasible partial Latin
    rectangle with first $j$ columns filled (with integers from 1 to $j$)
**block**[]: an auxiliary array used for backtracking
$\mathbf{B_k^j}$ : set of all blocks of size $k$ ending in $j$-th column; $c_{\min}(B_k^j)$
    denotes the minimal sum of corresponding costs among all
    blocks from $B_k^j$
**for** $j \leftarrow 2$ **to** $n$ **do**
    $value[j] = \min\{value[j-2] + c_{\min}(B_2^j),\ value[j-3] + c_{\min}(B_3^j)\}$;
    store in $block[j]$ for which block the minimum above is achieved;
**end**
$value[n]$ contains (only) an optimal value;
use $block[]$ to reconstruct the optimal solution;
```

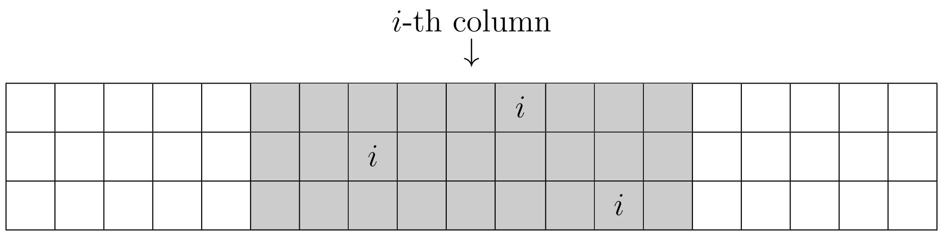

Figure 4.7: Representation of Corollary 4.14 for $p = 3$

Next we present Algorithm 4.2 (that can be found bellow) that solves the p-P3AP on layered Monge arrays. It is based on Corollary 4.14. The algorithm will start with the empty $p \times n$ Latin rectangle and will insert integers from 1 to n satisfying property (P').

Let us analyze the complexity of Algorithm 4.2. First let us observe the size of the state set F_i. Every state in F_i has the first $i - 2p + 1$ columns

Algorithm 4.2: p-P3AP on layered Monge arrays

```
initialize $F_0$ to be the set containing only the empty $p \times n$ Latin
rectangle;
for $i \leftarrow 1$ to $n$ do
    for all $L \in F_{i-1}$ do
        build the set $S_L$ of all Latin rectangles obtained from $L$ by
        adding the integer $i$ $p$ times to the columns $(i - 2p + 2)$ to
        $(i + 2p - 2)$ in all feasible ways;
        for all $X \in S_L$ do
            If the first $i - 2p + 2$ columns of $X$ are completely filled,
            and there is no Latin rectangle in $F_i$ with exactly the
            same positions filled and smaller corresponding cost sum,
            add $X$ to $F_i$ (and delete from $F_i$ the one with same
            positions filled and greater corresponding cost sum);
        end
    end
end
$F_n$ contains (only) an optimal solution;
```

completely filled, and the last $n - i - 2p + 2$ columns completely empty. A layout of filled and empty positions is unknown only in the $4p - 3$ columns in between, and they contain $p(2p - 1)$ integers, $2p - 1$ in each row. There is at most one state for every configuration of filled positions, hence the size of the state set is $|F_i| < \binom{4p-3}{2p-1}^p$, which is exponential function in p but does not depend on n.

In the second **for** loop of the algorithm we create S_L by inserting integer i p times, once to every row. We insert them only to the area of $4p - 3$ columns. Hence, the number of ways to do that is exponential in p but does not depend on n. Now it is clear that the complexity of Algorithm 4.2 is $O(f(p)n)$, where f is an exponential function. Furthermore, note that in every step i of the outer **for** loop we only use F_{i-1} to create F_i, hence the size of the memory needed depends only on p.

Corollary 4.15. *The p-P3AP for layered Monge cost arrays is fixed parameter tractable with respect to the parameter p.*

4.5 Special cases of the bottleneck-P3AP

In this section we investigate special cases of the bottleneck-P3AP. First note that the positive results of the previous sections of this chapter hold for the corresponding bottleneck variants.

Corollary 4.16. *For the bottleneck-*2*-P3AP with layered bottleneck Monge* $n \times n \times 2$ *cost array* C*, there always exists an optimal solution which represented as Latin rectangle decomposes into blocks with* 2 *or* 3 *columns.*

Proof. Analogous to the proof of Theorem 4.11. □

Corollary 4.17. *Let* I *be an instance of the bottleneck-*p*-P3AP with* $n \times n \times p$ *layered bottleneck Monge cost array* C*. There exists an optimal solution* X *for* I *such that the partial Latin square* L *that represents* X *has the following property:*

(Q) *Whenever the cell in* i*-th row and* j*-th column of* L $(L(i,j))$ *is filled, we have that* $|i-j| \le 2p-2$.

Proof. Analogous to the proof of Theorem 4.12. □

From the two corollaries above it follows that one can use modified versions of Algorithm 4.1 and Algorithm 4.2 (replace summation with max operator) to solve the bottleneck-p-P3AP for layered bottleneck Monge cost arrays. Hence the following corollary holds.

Corollary 4.18. *The bottleneck-*p*-P3AP for layered bottleneck Monge cost arrays is fixed parameter tractable with respect to the parameter* p.

We note that the complexity status of the bottleneck-P3AP on (layered) bottleneck Monge arrays is open. The technique used to prove the hardness result for the sum case does not work for the bottleneck case.

Open Problem 4.19. Determine the complexity of the bottleneck-P3AP on bottleneck Monge arrays and layered Bottleneck Monge arrays.

In the case of p-P3AP on bottleneck layered Monge arrays again there does not need to exist an optimal solution with a block structure when $p \ge 3$. Moreover, this holds even for the case of triply graded 0-1 arrays, see

Definition 4.25, which are special cases of bottleneck layered anti[1]-Monge arrays. The following $10 \times 10 \times 3$ array $C = (c_{ijk})$, given by $c_{ijk} = a^k_{ij}$, proves this fact:

$$A^1 = (a^1_{ij}) = \begin{pmatrix} 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&1\\ 0&0&0&0&0&0&0&1&1&1\\ 0&0&0&0&0&0&1&1&1&1\\ 0&0&0&0&0&1&1&1&1&1\\ 0&0&0&0&1&1&1&1&1&1\\ 0&0&0&0&1&1&1&1&1&1\\ 0&0&0&1&1&1&1&1&1&1 \end{pmatrix}$$

$$A^2 = (a^2_{ij}) = \begin{pmatrix} 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&1&1&1\\ 0&0&0&0&0&0&0&1&1&1\\ 0&0&0&0&0&1&1&1&1&1\\ 0&0&0&0&0&1&1&1&1&1\\ 0&0&0&0&1&1&1&1&1&1\\ 0&0&1&1&1&1&1&1&1&1\\ 0&0&1&1&1&1&1&1&1&1 \end{pmatrix}$$

$$A^3 = (a^3_{ij}) = \begin{pmatrix} 0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&0&0&1&1\\ 0&0&0&0&0&0&0&1&1&1\\ 0&0&0&0&0&0&1&1&1&1\\ 0&0&0&0&0&1&1&1&1&1\\ 0&0&0&0&1&1&1&1&1&1\\ 0&0&0&1&1&1&1&1&1&1\\ 0&0&1&1&1&1&1&1&1&1\\ 0&1&1&1&1&1&1&1&1&1 \end{pmatrix}.$$

[1]There is $\geq$ instead of $\leq$ in the definition of Monge array.

One solution of the bottleneck-3-P3AP on cost array C, expressed as Latin rectangle, is

8	7	10	9	5	6	2	4	1	3
9	10	6	8	7	4	5	1	3	2
10	9	8	7	6	5	4	3	2	1

,

which corresponds to one block of size 10. There is only one other optimal solution which is obtained by swapping integers 7 and 8 of first row. This alternative solution has the same "one block" property. It is easy to check that these two solutions are indeed the only two optimal solutions. The example above can be extended to arbitrary n and to $p \geq 3$ (just add additional irrelevant layers).

Next we investigate the case when cost arrays are the sum of three vectors, i.e.

$$c_{ijk} = a_i + b_j + d_k \tag{4.6}$$

for some real vectors $A = (a_i)$, $B = (b_i)$ and $D = (d_i)$. Note that the bottleneck-A3AP on this class of cost arrays is NP-hard. Namely, this follows directly from the NP-completeness of the *numerical three-dimensional matching* problem in [41]. We believe that the bottleneck-P3AP on the class of cost vectors of the form (4.6) is NP-Hard also.

Open Problem 4.20. Determine the complexity of the bottleneck-P3AP on cost arrays of the form (4.6).

Next we present one solvable more special case.

Proposition 4.21. *The bottleneck-P3AP with cost array $C = (c_{ijk})$, such that $c_{ijk} = a_i + b_j + d_k$, where a_i, b_i and d_i are arithmetic sequences, i.e.*

$$a_i = a_0 + i \cdot \alpha \quad b_i = b_0 + i \cdot \beta \quad d_i = d_0 + i \cdot \gamma,$$

for some α, β, γ, is polynomially solvable. Furthermore, assuming $\alpha \leq \beta \leq \gamma$, an optimal solution is given by:

$n-1$	$n-2$	$n-3$		2	1	n
$n-2$	$n-3$	$n-4$	$\cdots$	1	n	$n-1$
$n-3$	$n-4$	$n-5$		n	$n-1$	$n-2$
	$\vdots$		$\ddots$			
2	1	n		5	4	3
1	n	$n-1$	$\cdots$	4	3	2
n	$n-1$	$n-2$		3	2	1

, (4.7)

and the value of an optimal solution is $c_{1nn} = a_1 + b_n + d_n$ (*the upper rightmost cell in* (4.7)).

Proof. Consider first the rightmost column of a Latin square. It has to contain all integers from 1 to n. Corresponding cost values of that column have minimal bottleneck value if we arrange integers in such way that n is in the first row, $n-1$ is in the second, $n-2$ is in the third, and so on. The n in the first row will generate the greatest cost value ($a_1 + b_n + d_n$) of that column because $\gamma \geq \alpha$. Therefore $a_1 + b_n + d_n$ is a lower bound on the optimal solution value. Hence if we find a feasible solution (Latin square) with greatest cost element equal to $a_1 + b_n + d_n$, then that is an optimal solution. Latin square (4.7) is one such optimal solution. Namely, the cost value corresponding to n in the upper rightmost field of Latin square (4.7) is greater than the cost values corresponding to all other n's in (4.7). Also, for an arbitrary row the cost value corresponding to integer n is greater or equal than the cost values corresponding to all others integers in that row because $\gamma \geq \beta$. Hence, the cost value corresponding to n in the upper rightmost field is the greatest one. □

Lastly, we add that the bottleneck-p-P3AP on triply graded arrays (see Definition 4.25) can be solved using a variant of Algorithm 4.2. See Subsection 4.6.2 for more details.

4.6 Various other special cases

In this section we list some special case results that did not fit into the framework of the previous sections.

4.6.1 Maximization p-P3AP

It is straightforward to see that all results from Section 4.3 can be shown to be true for maximization versions of the problems. Clearly, feasible solutions will be reversed, i.e. in the Latin rectangle representation smaller integers will be on the right and larger on the left hand side. In the partial Latin square representation, integers will group around the antidiagonal. Also, the construction of the counterexample from Subsection 4.3.2 can be done analogously as follows. Q_3 remains the same. Q_2 is the same, except that the elements factored by 10 are on different positions, namely $p_{7,4,2} = 10a$ and $p_{2,9,2} = 10a^3$. Q_1 can be filled with 1's except $p_{5,7,1} = 100$. Then the unique optimal solution will be:

8	7	10	9	5	6	3	1	4	2
9	10	7	6	8	4	5	2	1	3
10	9	8	7	6	5	4	3	2	1

Corollary 4.22. *For the max-2-P3AP with a layered Monge $n \times n \times 2$ cost array C, there always exists an optimal solution which, represented as Latin rectangle, decomposes into reversed blocks with 2 or 3 columns.*

Corollary 4.23. *Let I be an instance of the max-p-P3AP with $n \times n \times p$ layered Monge cost array C. There exists an optimal solution X for I such that the partial Latin square L that represents X has the following property:*

(R) *Whenever the cell in i-th row and j-th column of L ($L(i, j)$) is filled, we have that $|i - n + j - 1| \leq 2p - 2$.*

Corollary 4.24. *The max-p-P3AP for layered Monge cost arrays is fixed parameter tractable with respect to the parameter p.*

4.6.2 Monotonicity property

Some hard combinatorial optimization problems become easy if the cost structure fulfills a monotonicity property. However, monotonicity does not help for the P3AP.

Definition 4.25. An $n_1 \times n_2 \times n_3$ cost array C is called *triply graded* if C is monotone increasing along all three lines of C. (That is, if $i_1 > i_2$, then $c_{i_1jk} \geq c_{i_2jk}$. Analogous for indices j and k.)

Theorem 4.26. *The P3AP and the p-P3AP remain NP-hard on the class of triply graded cost arrays.*

Proof. Let C be an $n \times n \times p$ cost array with $p \in \{2, \dots, n\}$. We will show that $C = (c_{ijk})$ can be turned into a triply graded cost array $\widetilde{C}$ such that the order of feasible solutions of the p-P3AP with respect to the objective function value does not change.

Let

$$m \geq |\max_{i,j,k} c_{ijk} - \min_{i,j,k} c_{ijk}|.$$

Define $\widetilde{c}_{ijk} = c_{ijk} + (i + j + k)m$. It follows from Observation 4.4 that the p-P3AP instances with cost arrays C and $\widetilde{C}$ are equivalent. Moreover, it can be easily seen that $\widetilde{C}$ is monotone increasing along all lines and hence triply graded. □

Note that the same approach also works if the direction of monotonicity is not the same in all three coordinate directions.

In the case of triply graded arrays the sum and bottleneck versions of the p-P3AP behave differently. Namely, triply graded arrays are bottleneck anti-Monge arrays, hence the bottleneck-p-P3AP can be solved by a bottleneck variant of Algorithm 4.2 where the order of columns is reversed.

4.6.3 The P3AP on distribution arrays generated by a single nonzero density element

In this subsection we investigate the solvability of the P3AP with distribution cost arrays, with respect to the number of nonzero density elements. First we state a classical result concerning Latin square completions.

Theorem 4.27 (Ryser [73])**.** *Let T be an $r \times s$ Latin rectangle filled with the integers $1, 2, \dots, n$. Let $N(i)$ denote the number of times that the integer i occurs in T. A necessary and sufficient condition in order that T may be extended to an $n \times n$ Latin square is that for each $i = 1, 2, \dots, n$,*

$$N(i) \geq r + s - n.$$

Proposition 4.28. *For the P3AP with a distribution cost array*

$$c_{ijk} = \sum_{i'=1}^{i} \sum_{j'=1}^{j} \sum_{k'=1}^{k} -p_{i'j'k'},$$

where the density array $P = (p_{ijk})$ contains a single 1 entry and all other entries are 0, there exists a polynomial algorithm for finding optimal solution.

Proof. Let $p_{ijk} = 1$ be the one nonzero density element. Then the problem reduces to finding a Latin square such that its $(n-i+1) \times (n-j+1)$ bottom right Latin rectangle contains as many integers greater or equal than k as possible. Let us denote by r and s the number of rows and columns of such bottom right Latin rectangle, i.e. $r := n - i + 1$, and $s := n - j + 1$. We can assume that $r \geq s$, otherwise rotate the Latin square by 90 degrees and proceed analogously.

One way of filling the $r \times s$ rectangle such that it has minimal sum of the corresponding cost values, with only constraint being that same integers do not appear in the same rows or columns, can be done as follows:

$T :=$

			n-s+2	⋯	n-1	n	n-r+1
			n-s+3	⋯	n	n-r+1	n-r+2
			n-s+4	⋯	n-r+1	n-r+2	n-r+3
			⋮			⋮	⋮
			n-s-1	⋯	n-4	n-3	n-2
			n-s	⋯	n-3	n-2	n-1
			n-s+1	⋯	n-2	n-1	n

.

The optimality of such rectangle T does not depend on the value of k (in $p_{ijk} = 1$) but it may not satisfy Theorem 4.27, i.e. it could be not extendable to an $n \times n$ Latin square. We will modify Latin rectangle T so that it remains optimal and that it satisfies Theorem 4.27.

In T there are r different integers, and each of them appears s times. We need to add remaining $n - r$ integers $N(i) = r + s - n$ times. Integers that are already in T appear there s times, therefore by Theorem 4.27 $s - (r + s - n) = n - r$ of them can be removed. Hence, if we remove smallest $r + s - n$ integers from T $n - r$ times, and replace them by the missing $n-r$ integers $r+s-n$ times, we will get a Latin rectangle that can be extended to a Latin square, and when extended it will be an optimal solution.

We can do the exchange as follows: Insert integer 1 to the first column of T in place of integer $n - r + 1$. Insert other 1's in place of $n - r + 2$, $n - r + 3$, $n - r + 4$ and so on, in such way that the next insertion is two columns right and one row above. The row "above" the first row is the last row. After making an insertion to one of the last two columns, jump to the second column to find the next integer to exchange. Insert 1's until all $r + s - n$ 1's are inserted. In doing so, all 1's will be in different columns but also in different rows, because if $r < n$, then $N(i) = r+s-n < s$ (look at the row difference between two 1's in first two columns.) Now insert the next integer, i.e. 2, $s + r - n$ times by inserting it into the cell one row above and one column to the right of the previously inserted integers, i.e. 1's. Proceed like this until all $n-r$ integers are inserted. It is obvious that in this way we get an optimal extensible Latin rectangle, which we denote by T'. In Figure 4.8, a transformation from T to T' is depicted.

Next, we will describe how to extend Latin rectangle T' to a Latin square. This extension is also a sufficiency proof of Theorem 4.27, and it is derived from it, see [73].

We form a 0-1 $r \times n$ matrix A in the following way. Let A contain 0 in the i-th row and the j-th column if and only if there is integer j in the i-th row of T'. Note that A has exactly $\ell := n - s$ ones in each row. Denote by $M(i)$ the number of 1's in the i-th column. $N(i) = r - M(i) \geq r + s - n$, hence $M(i) \leq \ell$ for all i. Since T' is an $r \times s$ Latin rectangle, it follows

$p_{3,4,k} = 1$
$n = 11$
$r = 9$
$s = 8$
$N(i) = 6$
$n - r = 2$

red $\longrightarrow 1$
blue $\longrightarrow 2$

			5	6	**7**	**8**	9	10	11	3
			6	**7**	8	9	10	11	3	4
			7	8	9	10	11	3	4	5
			8	9	10	11	3	4	5	**6**
			9	10	11	3	4	**5**	**6**	7
			10	11	3	**4**	**5**	6	7	8
			11	**3**	**4**	5	6	7	8	9
			3	4	5	6	7	8	9	10
			4	5	6	7	**8**	9	10	11

Figure 4.8: Transforming T into T'

that $N(i) \leq s$. Hence $\ell - (n-r) \leq M(i)$. Finally,

$$\ell - (n-r) \leq M(i) \leq \ell.$$

Now we will add $n-r$ rows to A so that A becomes an $n \times n$ matrix with exactly ℓ 1's in every row and column. Let t denote the number of columns of A with $M(i) < \ell$. Then $n-t$ denotes the number of columns of A with $M(i) = \ell$, and consequently

$$\ell r = \sum_{i=1}^{n} M(i) \geq (n-t)\ell + (\ell - (n-r))t.$$

Thus, $\ell(r-n) \geq t(r-n)$, hence $t \geq \ell$.

Next, let p denote the number of columns of A with $M(i) = \ell - (n-r)$. Then $n-p$ is the number of columns with $M(i) > \ell - (n-r)$. Consequently,

$$\ell r = \sum_{i=1}^{n} M(i) \leq p(\ell - (n-r)) + (n-p)\ell,$$

hence $\ell(r-n) \leq p(r-n)$ and $p \leq \ell$.

We now add to A a row consisting of ℓ 1's and $n-\ell$ 0's. Since $t \geq \ell$, there are at least ℓ positions where 1's may be inserted so that the resulting $(r+1)$-rowed matrix will have at most ℓ 1's in each column. Moreover, since

$p \leq \ell$, the 1's may be inserted to all columns with $M(i) = \ell - (n - r)$. In the resulting $(r+1)$-rowed matrix, let $M'(i)$ denote the number of 1's in the i-th column. Because of the structure of the added row, it is clear that

$$\ell - (n - (r+1)) \leq M'(i) \leq \ell.$$

The process may be continued inductively, and the resulting square matrix A' possesses ℓ 1's in each row and column. Matrix A' satisfies Hall's marriage theorem. Namely, for any r rows of A' there are at least r columns containing 1. Assume that there are only $r-1$ columns containing 1. Because every column in A' contains ℓ 1's, the chosen r rows would contain at most $(r-1)\ell$ 1's, but any r rows of A' contain $r\ell$ 1's, contradiction.

Now construct one matching $m := (t_1, t_2, \dots, t_n)$ from A', where t_i denotes the column of chosen 1 from the row i. Now take the last r elements of m and add it as a column to T'. After removing the chosen 1's from A', it still satisfies Hall's marriage theorem, so we can proceed inductively until we extend T' to an $r \times n$ Latin rectangle that we denote by T''.

Now all we need to do is to add $n-r$ rows to T'' to get a Latin square. Let D be a $n \times n$ 1-0 matrix containing 1 in the i-th row and the j-th column if and only if j-th column of T'' does not contain integer i. It is easy to see that D contains exactly $n-r$ 1's in every row and column. Therefore it satisfies Hall's marriage theorem, hence we can inductively extract matchings from D and add it as a row to T''. This concludes the algorithm. □

4.6.4 Greedily solvable instances with layered Monge cost arrays

Consider the following greedy algorithm for the p-P3AP on a cost array C. First find a matching M_1 that solves the linear assignment problem (LAP) on the first layer of C (i.e. on the matrix c_{ij1} for $i, j = 1, \dots, n$). Then find a matching M_2 that solves the LAP on the second layer of C with additional constraint being that M_1 and M_2 are disjoint. Then find a third matching disjoint to the previous two in the analogous way. Continue in this manner until obtaining p disjoint matchings that determine the solution of the p-P3AP.

In this subsection we want to know for which layered Monge arrays p-P3AP is solved by the greedy algorithm. In the light of Proposition 4.2, w.l.o.g. we can consider only Monge matrices that are distribution matrices.

We answer this question for $p = 2$.

Proposition 4.29. *Let $C = (c_{ijk})$ be an $n \times n \times 2$ layered Monge array such that $D^1 = (d^1_{ij})$ and $D^2 = (d^2_{ij})$ with $d^1_{ii} > 0$ for $i = 1, \dots, n$ are the density matrices of the corresponding Monge matrix layers of C (i.e. $c_{ijk} = -\sum_{r=1}^{i}\sum_{s=1}^{j} d^k_{rs}$). Then the 2-P3AP on cost array C can be solved to optimality by the greedy algorithm if*

$$d^1_{ii} \geq \max\{d^2_{ii}, d^2_{i-1,i}, d^2_{i,i+1}\} \quad \textit{for all } i \in \{1, 2, ..., n-1\}. \tag{4.8}$$

Proof. By Theorem 4.11 there exists an optimal solution that represented as Latin rectangle L consists only of blocks of size 2 and 3. From the condition $d^1_{ii} > 0$, for $i = 1, \dots, n$, it follows that the Latin rectangle representation of every solution obtained by the greedy algorithm will have

$$\begin{array}{|c|c|c|c|} \hline 1 & 2 & \quad\cdots\quad & n \\ \hline \end{array} \tag{4.9}$$

as its first row. Moreover, the solution obtained by the greedy algorithm will have the objective value less or equal to the objective value of any Latin rectangle with the first row being (4.9). Hence, by checking when there is no block of size 2 or 3 that has the corresponding cost sum strictly smaller than every block consisting of the same integers with the first row of the form $\begin{array}{|c|c|c|} \hline j & j+1 & j+2 \\ \hline \end{array}$ or $\begin{array}{|c|c|} \hline j & j+1 \\ \hline \end{array}$, we get the sufficient conditions for the optimality of the greedy algorithm. In doing so we get (4.8). $\square$

For $p \geq 3$ the problem gets significantly harder. Even for arbitrarily large $a > 0$ the condition $d^k_{i,j} > a \cdot d^{k+1}_{i,j}$ for all i, j, k does not force optimality of the greedy algorithm. Namely, the difference of costs of matchings from different layers are not important, but the difference of costs of matchings in the same layer. Furthermore, if more than one matching is the feasible outcome of a step of the greedy algorithm, analysis gets out of control.

Open Problem 4.30. Find other classes of cost arrays for which the p-P3AP is solved to optimality by the greedy algorithm.

Chapter 5

The constant objective value property for combinatorial optimization problems

The topic of this chapter is the following. Given a combinatorial optimization problem, we aim at characterizing the set of all instances for which every feasible solution has the same objective value.

Our central result deals with multidimensional assignment problems. We show that for the axial and for the planar d-dimensional assignment problem instances with constant objective value property are characterized by sum-decomposable arrays. We provide a counterexample to show that the result does not carry over to general d-dimensional assignment problems.

Our result for the axial d-dimensional assignment problems can be shown to carry over to the axial d-dimensional transportation problem. Moreover, we obtain characterizations when the constant objective value property holds for the minimum spanning tree problem, the shortest path problem and the minimum weight maximum cardinality matching problem.

The results of this chapter can be found in [28].

5.1 Constant objective value property

In this chapter we will deal with the following type of combinatorial optimization problems. We are given a ground set $E = \{1, \ldots, n\}$, a real cost vector $C = (c(1), \dots, c(n))$ and a set of feasible solutions $\mathcal{F} \subseteq 2^{\{1,\dots,n\}}$.

The *objective value* of a feasible solution $F \in \mathcal{F}$ is given by the so-called *sum objective function*

$$c(F) := \sum_{i \in F} c(i).$$

The goal is to find a feasible solution F^* such that $c(F^*)$ is minimal.

The traveling salesman problem, the linear assignment problems, the shortest path problem, Lawler's quadratic assignment problem and many other well-known combinatorial optimization problems fall into the class of combinatorial optimization problems described above.

Definition 5.1. We say that an instance of a combinatorial optimization problem has the constant objective value property (COVP) if every feasible solution has the same objective value.

Our goal is to characterize the set of instances with the COVP, or in other words, the space of all cost vectors for which every feasible solution has the same objective value, for various combinatorial optimization problems.

The constant objective value property is closely connected to the notion of *admissible transformations* introduced in 1971 by Vo-Khac [50].

Definition 5.2. A transformation T of the cost vector C to the new cost vector $\tilde{C} = (\tilde{c}(1), \tilde{c}(2), \dots, \tilde{c}(n))$ is called admissible with index $z(T)$, if

$$c(F) = \tilde{c}(F) + z(T) \quad \text{for all } F \in \mathcal{F}.$$

Note that admissible transformations preserve the relative order of the objective values of all feasible solutions. It is well known that admissible transformations can be used as optimality criterion and to obtain lower bounds which are useful for hard combinatorial optimization problems. Namely, consider the combinatorial optimization problem $\min_{F \in \mathcal{F}} c(F)$. Let T be an admissible transformation with index $z(T)$ from the original cost vector C to the new cost vector $\tilde{C}$ such that there exists a feasible solution F^* with the following properties:

(i) $\tilde{c}(i) \geq 0 \quad \text{for all } i \in \{1, \dots, n\}$,

(ii) $\tilde{c}(F^*) = 0$.

Then F^* is an optimal solution with objective value $z(T)$. If the condition (ii) is not satisfied or we cannot prove that it holds, then $z(T)$ gives a lower bound.

For the class of combinatorial optimization problems with sum objective function there is a one-to-one correspondence between admissible transformations that transform the cost vector $(c(1), \ldots, c(n))$ to $(\tilde{c}(1), \ldots, \tilde{c}(n))$, and cost vectors $B = (b(1), b(2), \ldots, b(n))$ that fulfill the COVP. The correspondence is obtained by $c(i) = \tilde{c}(i) + b(i)$ for all i. Then the index of the corresponding admissible transformation is $z(T) = \sum_{i \in F} b(i)$ for any $F \in \mathcal{F}$. The correspondence between the COVP and admissible transformations provides a further source of motivation for investigating COVP characterizations.

The notion of admissible transformations can be generalized to the algebraic setting and applied to a wider class of combinatorial optimization problems, including the case of bottleneck objective functions, see [24]. Note, however, that for the bottleneck objective function, which is given by $c(F) = \max_{i \in F} c(i)$, there is no one-to-one correspondence between the COVP and admissible transformations.

Berenguer [14] characterized the set of all admissible transformations for the travelling salesman problem (TSP) and the multiple salesmen version. All admissible transformations for the TSP are obtained by adding real values to rows and columns of the distance matrix. In view of the correspondence mentioned above this result can be rephrased as a result on the COVP for the TSP as follows (this has been noted already by Gilmore, Lawler and Shmoys [43]).

An $n \times n$ real matrix $C = (c_{ij})$ is called *sum matrix* if there exist two real n-dimensional vectors $U = (u_i)$ and $V = (v_i)$ such that

$$c_{ij} = u_i + v_j \qquad \text{for all } \ i, j \in \{1, \ldots, n\}. \tag{5.1}$$

Theorem 5.3 (Berenguer [14], Gilmore et al. [43])**.** *The TSP instance with a cost matrix $C = (c_{ij})$ has the COVP if and only if C is a sum matrix.*

For the TSP the diagonal entries of C do not play a role and can be ignored. Berenguer's proof works for the linear assignment problem as well, i.e. an instance of the linear assignment problem with cost matrix

$C = (c_{ij})$ has the COVP if and only if C is a sum matrix.

Some classes of admissible transformations for different types of assignment problems are listed by Burkard [19]. However, no COVP characterizations are provided.

We remark that there is a simpler way to prove the COVP characterization for the linear assignment problem mentioned above by making use of the LP-duality and the complementary slackness condition. Since for each pair (i, j) there exists an assignment which assigns i to j (i.e. the primal assignment variable x_{ij} is 1), all dual constraints need to be fulfilled with equality which is equivalent to the condition (5.1) (note that the vectors U and V contain the dual variables).

The structure of the rest of the chapter is as follows. In Section 5.2 we investigate the problem of characterizing the instances with the COVP for multidimensional assignment problems. We show that for the multidimensional axial and planar case the cost arrays with the COVP are precisely the class of sum-decomposable arrays which can be represented as sums of lower dimensional arrays of appropriate dimension (for the precise definition see Section 5.2). We furthermore provide a counterexample which shows that sum-decomposability is not necessarily required for the COVP to hold for general multidimensional assignment problems. In Section 5.3 the result for the axial d-dimensional assignment problem is carried over to the axial d-dimensional transportation problem. Finally, in Section 5.4 we deal with COVP characterizations for the minimum spanning tree problem, the shortest path problem and the minimum weight maximum cardinality matching problem.

5.2 The COVP for d-dimensional assignment problems

Berenguer's result for the classical linear assignment problem motivated us to ask for COVP characterizations for multidimensional assignment problems (d, s)-AP. See Section 2.4 for the definition and introduction to multidimensional assignment problems (d, s)-AP.

5.2.1 Sum-decomposable arrays

In this subsection we investigate the vector spaces of *sum-decomposable arrays*. These will occur as solutions of various COVP characterizations. Sum-decomposable arrays generalize the concept of sum matrices to higher dimensions.

Informally, a d-dimensional $n \times n \times \cdots \times n$ real array C is sum-decomposable with parameters d and s (and size n) if C can be obtained as a sum of $\binom{d}{s}$ s-dimensional arrays, one for each subset of $\{1, \ldots, d\}$ of size s. For example, in the case $d = 3$ and $s = 2$, $C = (c_{ijk})$ is sum-decomposable if there exist three two-dimensional real arrays $A = (a_{ij})$, $B = (b_{ij})$ and $D = (d_{ij})$ such that $c_{ijk} = a_{ij} + b_{ik} + d_{jk}$. A formal definition follows.

Definition 5.4. Let n, d and s be integers such that $d > s > 0$ and $n > 1$. Let $\mathcal{Q}_s = \{Q \colon Q \subset \{1, \ldots, d\}, |Q| = s\}$. Then the d-dimensional $n \times n \times \cdots \times n$ real array C is called *sum-decomposable with parameters d and s and size n* if there exist $\binom{d}{s}$ s-dimensional $n \times n \times \cdots \times n$ real arrays $A^Q = (a^Q(j_1, \ldots, j_s))$, one for each $Q \in \mathcal{Q}_s$, such that

$$c(i_1, i_2, \ldots, i_d) = \sum_{Q \in \mathcal{Q}_s} a^Q(h_Q(i_1, i_2, \ldots, i_d))$$

where $h_Q(i_1, i_2, \ldots, i_d)$ denotes the s-tuple associated with Q, i.e. $h_Q(i_1, i_2, \ldots, i_d) = (i_{q_1}, \ldots, i_{q_s})$ for $Q = \{q_1, \ldots, q_s\}$, $q_1 < q_2 < \cdots < q_s$.

We denote the vector space of all sum-decomposable real arrays of size n with parameters d and s by $\mathrm{SAVS}(d, s, n)$.

For $Q = \{j_1, j_2, \ldots, j_s\} \in \mathcal{Q}_s$ let V_Q denote the vector space of all d-dimensional $n \times n \times \cdots \times n$ arrays $C = (c(i_1, i_2, \ldots, i_d))$ for which there exists a mapping $f \colon \{1, 2, \ldots, n\}^s \mapsto \mathbb{R}$ with $c(i_1, i_2, \ldots, i_d) = f(i_{j_1}, i_{j_2}, \ldots, i_{j_s})$. In other words, the value $c(i_1, i_2, \ldots, i_d)$ depends only on the s indices from the set Q and not on all d indices. Let $Q_1, Q_2, \ldots, Q_{\binom{d}{s}}$ be such that $\mathcal{Q}_s = \{Q_1, \ldots, Q_{\binom{d}{s}}\}$. Note that

$$\mathrm{SAVS}(d, s, n) = V_{Q_1} + V_{Q_2} + \cdots + V_{Q_{\binom{d}{s}}}. \tag{5.2}$$

We will use the following proposition, that can be found in [65, Prop. 7.1 of Chap. 1, p. 15], to prove that a variant of inclusion-exclusion principle holds for the vector subspaces V_{Q_i}. Recall that the distributivity with respect to sum and intersection does not hold for arbitrarily vector subspaces, i.e. it is not true that $V_1 \cap (V_2 + V_3) = (V_1 \cap V_2) + (V_1 \cap V_3)$ holds for all vector spaces V_1, V_2, V_3.

Proposition 5.5. *Let W be a vector space and $V_1, V_2, \dots, V_n \subset W$ be a collection of its subspaces. Then the following conditions are equivalent:*

(i) The collection $V_1, V_2, \dots, V_n$ is distributive with respect to the operations of sum and intersection.

(ii) There exists a basis $\{w_\alpha : \alpha \in A\}$ of the vector space W such that each of the subspaces V_i is the linear span of a set of vectors w_α.

Now we are ready to prove the following proposition.

Proposition 5.6. *Let* $\mathrm{SAVS}(d, s, n)$ *be expressed as in* (5.2). *Then we have*

(i) $\dim(V_Q) = n^s$ *for all $Q \in \mathcal{Q}_s$,*

(ii) $\dim(\cap_{i \in I} V_{Q_i}) = n^{|\cap_{i \in I} Q_i|}$ *for all $I \subset \{1, \dots, \binom{d}{s}\}$,*

(iii) $$\dim(\mathrm{SAVS}(d, s, n)) = \dim\left(\sum_{i=1}^{\binom{d}{s}} V_{Q_i}\right) =$$
$$\sum_{k=1}^{\binom{d}{s}} (-1)^{k+1} \left(\sum_{1 \le i_1 < \dots < i_k \le \binom{d}{s}} \dim\left(V_{Q_{i_1}} \cap \dots \cap V_{Q_{i_k}}\right)\right),$$

(iv) $\dim(\mathrm{SAVS}(d, d-1, n) = n^d - (n-1)^d$,

$\dim(\mathrm{SAVS}(d, 1, n) = dn - d + 1$.

Proof. Ad (i): Follows directly from the definition of V_Q.

Ad (ii): Let us start with $|I| = 2$ and consider $J = \{j_1, j_2, \dots, j_s\}$, $K = \{k_1, k_2, \dots, k_s\}$ from $\mathcal{Q}_s$. Further, let C and E be two arrays from V_J and V_K, respectively. Hence, $c(i_1, i_2, \dots, i_d) = f(i_{j_1}, i_{j_2}, \dots, i_{j_s})$ and $e(i_1, i_2, \dots, i_d) = g(i_{k_1}, i_{k_2}, \dots, i_{k_s})$ for some $f, g\colon \{1, 2, \dots, n\}^s \mapsto \mathbb{R}$. Then for every $A = (a(i_1, \dots, i_d)) \in V_J \cap V_K$ we have that $a(i_1, i_2, \dots, i_d) = t(i_{q_1}, \dots, i_{q_{|J \cap K|}})$, $q_i \in J \cap K$, for some $t\colon \{1, 2, \dots, n\}^{|J \cap K|} \mapsto \mathbb{R}$. Hence,

$\dim(V_J \cap V_K) = n^{|J \cap K|}$. The case $|I| \geq 3$ follows by an inductive argument. This settles (ii).

Ad (iii): The case of two vector spaces involved in the sum follows from the fact that

$$\dim(V_1 + V_2) = \dim(V_1) + \dim(V_2) - \dim(V_1 \cap V_2) \tag{5.3}$$

holds for any two subspaces V_1 and V_2 of a vector space. Next we settle the case of three vector spaces, so let $Q_i, Q_j, Q_\ell \in \mathcal{Q}_s$. The general case then follows by induction. Note that (5.3) implies that

$$\begin{aligned}\dim\left(V_{Q_i} + V_{Q_j} + V_{Q_\ell}\right) = \\ \dim\left(V_{Q_i}\right) + \dim\left(V_{Q_j} + V_{Q_\ell}\right) - \dim\left(V_{Q_i} \cap \left(V_{Q_j} + V_{Q_\ell}\right)\right).\end{aligned} \tag{5.4}$$

To proceed further it suffices to prove that the subspaces V_{Q_i} are distributive with respect to sum and intersection, i.e. that

$$V_{Q_i} \cap \left(V_{Q_j} + V_{Q_\ell}\right) = \left(V_{Q_i} \cap V_{Q_j}\right) + \left(V_{Q_i} \cap V_{Q_\ell}\right) \tag{5.5}$$

holds. It is easy to check that using (5.5) and (5.3) in (5.4) above leads to the claim (iii) for three vector spaces. So, it only remains to show that the distributivity property (5.5) holds. To that end, we construct bases for the vector spaces V_Q for $Q \in \mathcal{Q}_s$ and for the vector space V of all d-dimensional $n \times \cdots \times n$ real arrays. The distributivity then follows from Proposition 5.5. Namely, call an array from V *elementary* if a single entry is 1 and all other entries are 0. It is easy to see that the set of n^d d-dimensional $n \times n \times \cdots \times n$ elementary arrays forms a basis for V. Next we construct a basis for the subspace V_Q where $Q = \{j_1, j_2, \ldots, j_s\}$. Let $A_{k_1,\ldots,k_s}$ be the 0-1 d-dimensional array such that the entry at position $(i_1, \ldots, i_d)$ is 1 if $i_{j_1} = k_1$, …, $i_{j_s} = k_s$ and 0 otherwise. Then the set of arrays $\{A_{k_1,\ldots,k_s} \colon (k_1, \ldots, k_s) \in \{1, \ldots, n\}^s\}$ forms a basis for V_Q. Note that every element of the basis for V_Q can be written as a linear combination of elementary arrays.

Ad (iv): Note that any intersection of ℓ distinct subsets of $\{1, 2, \ldots, d\}$

has cardinality $d-\ell$. Hence, from (ii) and (iii) it follows that

$$\dim(\mathrm{SAVS}(d,d-1,n)) = \sum_{i=1}^{d}(-1)^{i+1}\binom{d}{i}n^{d-i},$$

which is equal to $n^d-(n-1)^d$ by the binomial theorem. Since the intersection of any distinct one-element sets is empty it follows that

$$\dim(\mathrm{SAVS}(d,1,n)) = dn-d+1.$$

□

5.2.2 The COVP for the axial case: $(d,1)$-AP

Now we turn to the problem of characterizing the instances of the axial d-dimensional assignment problem with the constant objective value property (COVP).

Theorem 5.7. *An instance of the $(d,1)$-AP with cost array C has the COVP if and only if C is a sum-decomposable array with parameters d and 1.*

Proof. Note that one direction follows immediately, i.e. if C is the sum of d vectors, then every feasible solution has the same objective value. Conversely, assume that every feasible solution has the same objective value. For integers $i_1,i_2,\dots,i_d\in\{2,3,\dots,n\}$ consider the following d pairs of d-tuples:

$$\begin{array}{ll}
(1,1,\dots,1), & (i_1,i_2,\dots,i_d)\\
(i_1,1,\dots,1), & (1,i_2,\dots,i_d)\\
(1,i_2,1,\dots,1), & (i_1,1,i_3,\dots,i_d)\\
\quad\vdots & \\
(1,\dots,1,i_d), & (i_1,\dots,i_{d-1},1).
\end{array}$$

There exists a set of $n-2$ d-tuples which completes each of these pairs to a feasible solution; for example the set $\{(k_1^j,k_2^j,\dots,k_d^j)\colon j=2,\dots,n-1,\ k_l^j=$

j if $j < i_l$ and $k_l^j = j+1$ otherwise, $l = 1, \dots d\}$. By assumption we have

$$\begin{aligned} c(i_1, i_2, \dots, i_d) &= c(i_1, 1, \dots, 1) + c(1, i_2, \dots, i_d) - c(1, \dots, 1) && (5.6) \\ &= c(1, i_2, 1, \dots, 1) + c(i_1, 1, i_3, \dots, i_d) - c(1, \dots, 1) && (5.7) \\ &\vdots \\ &= c(1, \dots, 1, i_d) + c(i_1, \dots, i_{d-1}, 1) - c(1, \dots, 1). \end{aligned}$$

Due to (5.6) there exist a vector $V_1 = (v_1(i))$ and a $(d-1)$-dimensional array $G_1 = (g_1(i_1, \dots, i_{d-1}))$ such that $c(i_1, i_2, \dots, i_d) = v_1(i_1) + g_1(i_2, \dots, i_d)$. Analogously, from (5.7) it follows that there exists a vector V_2 and a $(d-1)$-dimensional array G_2 such that $c(i_1, i_2, \dots, i_d) = v_2(i_2) + g_2(i_1, i_3, \dots, i_d)$. Hence, $c(i_1, i_2, \dots, i_d) = v_1(i_1) + v_2(i_2) + g_{1,2}(i_3, \dots, i_d)$ for some $(d-2)$-dimensional array $G_{1,2} = (g_{1,2}(i_1, \dots, i_{d-2}))$. Using the remaining equations in an analogous manner we finally obtain that C is the sum of d vectors, i.e.

$$c(i_1, i_2, \dots, i_d) = v_1(i_1) + v_2(i_2) + \dots + v_d(i_d),$$

where the vectors $V_k = (v_k(i))$ can be chosen as follows:

$$\begin{aligned} v_1(i) &= c(i, 1, \dots, 1) - \frac{d-1}{d} c(1, 1, \dots, 1), \\ &\vdots \\ v_d(i) &= c(1, \dots, 1, i) - \frac{d-1}{d} c(1, 1, \dots, 1). \end{aligned}$$

□

5.2.3 The COVP for the planar case: $(d, d-1)$-AP

We now turn to the planar case. Note that there are exactly two feasible solutions of the $(d, d-1)$-AP when $n = 2$.

Definition 5.8. We say that an instance of the $(d, d-1)$-AP with cost array C has property P_2 if for every $2 \times 2 \times \dots \times 2$ sub-array of C, which is obtained by restricting the index sets to $\{1, i_1\} \times \{1, i_2\} \times \dots \times \{1, i_d\}$, the two feasible solutions on the resulting subproblem of size 2 have the same objective value.

Property P_2 and sum-decomposable cost arrays for the $(d, d-1)$-AP are related in the following way.

Lemma 5.9. *Let I be an instance of the $(d, d-1)$-AP with cost array C. If I has property P_2, then C is a sum-decomposable array with parameters d and $d-1$.*

Proof. Consider the $2 \times 2 \times \cdots \times 2$ subarray D_2 of C obtained by restricting index sets to $\{1, i_1\} \times \{1, i_2\} \times \cdots \times \{1, i_d\}$ with $i_j \in \{2, \ldots, n\}$ for $j = 1, \ldots, d$. By exploiting the fact that the two feasible solutions for the subarray D_2 have the same objective value we get that

$$c(i_1, i_2, \ldots . i_d) = \sum_{x \in I_1} c(x) - \sum_{x \in I_2} c(x) + \cdots + (-1)^{d+1} \sum_{x \in I_d} c(x), \tag{5.8}$$

where I_i is the set of all d-tuples from $\{1, i_1\} \times \{1, i_2\} \times \cdots \times \{1, i_d\}$ with exactly i ones. Then from (5.8) it follows that C can be expressed as the sum of d $(d-1)$-dimensional arrays $A_j = (a_j(i_1, \ldots, i_{d-1}))$, $j = 1, \ldots, d$, defined by

$$\begin{aligned}
a_1(i_2, i_3, \ldots, i_d) &= \sum_{x \in I_1^1} c(x) - \frac{1}{2} \sum_{x \in I_2^1} c(x) + \cdots + (-1)^{d+1} \frac{1}{d} \sum_{x \in I_d^1} c(x) \\
a_2(i_1, i_3, \ldots, i_d) &= \sum_{x \in I_1^2} c(x) - \frac{1}{2} \sum_{x \in I_2^2} c(x) + \cdots + (-1)^{d+1} \frac{1}{d} \sum_{x \in I_d^2} c(x) \\
&\vdots \\
a_d(i_1, i_2, \ldots, i_{d-1}) &= \sum_{x \in I_1^d} c(x) - \frac{1}{2} \sum_{x \in I_2^d} c(x) + \cdots + (-1)^{d+1} \frac{1}{d} \sum_{x \in I_d^d} c(x),
\end{aligned}$$

where I_i^k is the set of all d-tuples from $\{1, i_1\} \times \{1, i_2\} \times \cdots \times \{1, i_d\}$ with exactly i ones, one of which is on the k-th coordinate. □

The following result relates property P_2 and the COVP.

Proposition 5.10. *Every instance of the $(d, d-1)$-AP with cost array C with $n \neq 3$ that has the COVP, also has property P_2.*

Proof. We will prove that both feasible solutions of the $(d, d-1)$-AP on the sub-array of C with indices $\{1, 2\} \times \{1, 2\} \times \cdots \times \{1, 2\}$ have the same objective value; the general case can be shown analogously. When $n = 2$, this is trivially true. Assume $n \geq 4$. We will build two different feasible solutions F_1^d and F_2^d for the $(d, d-1)$-AP that satisfy the following property: F_1^d and F_2^d both contain a feasible solution of the $(d, d-1)$-AP on the subproblem induced by the index set $\{1, 2\}^d$, and all other elements of these two solutions are the same. The existence of such F_1^d and F_2^d completes the proof. Namely, by assumption the objective values of F_1^d and F_2^d are equal, hence (5.8) holds.

Next we explain how F_1^d and F_2^d can be constructed recursively from a feasible solution of the $(d-1, d-2)$-AP, which we denote by F^{d-1}, which also contains a feasible solution on the subproblem of size 2 induced by the index set $\{1, 2\}^{d-1}$. We define F_j^d, $j = 1, 2$ as follows:

$$F_j^d = \{(i, a_1, a_2, \ldots, a_{d-2}, \phi_i^j(a_{d-1}))\colon (a_1, \ldots, a_{d-1}) \in F^{d-1}, i = 1, \ldots, n\},$$

where n permutations ϕ_i^j, $i = 1, \ldots, n$, are chosen to be mutually disjoint (recall that two permutations α and β are disjoint if $\alpha(i) \neq \beta(i)$ for all i). Furthermore, for every i we choose ϕ_i^1 and ϕ_i^2 such that they coincide except for $\phi_1^1(1) = 1$, $\phi_1^1(2) = 2$, $\phi_2^1(1) = 2$, $\phi_2^1(2) = 1$, in contrast to $\phi_1^2(1) = 2$, $\phi_1^2(2) = 1$, $\phi_2^2(1) = 1$, $\phi_2^2(2) = 2$. To show that such two sets of permutations (for $j = 1$ and $j = 2$) exist, we represent them as two $n \times n$ Latin squares. For $j = 1, 2$, let the j-th table contain the integer $\phi_r^j(s)$ in the row r and column s. The resulting tables will be two Latin squares of order n which are identical except in the 2×2 upper-left corner. That corner is filled with two different Latin squares of order 2, respectively. It is well known that for $n \geq 4$ such Latin squares exist, see [73]. From Observation 2.7 we get that F_j^d are indeed feasible solutions. $\square$

The approach we followed in the proof of Proposition 5.10 did not serve us to cover the case $n = 3$ and $d \geq 5$. Using a linear algebra approach we were able to cover this case as well and hence to prove the following COVP characterization for the $(d, d-1)$-AP.

Theorem 5.11. *An instance of the $(d, d-1)$-AP with cost array C has the COVP if and only if C is a sum-decomposable array with parameters d and $d-1$.*

Proof. If the cost array C is sum-decomposable, then it is straightforward to see that every feasible solution has the same objective value.

Conversely, assume that every feasible solution has the same objective value. For the case $n \neq 3$ the statement follows from Proposition 5.10 and Lemma 5.9. For the remaining case $n = 3$ we make use of the same technique that has been used in [43, 57] to obtain a COVP characterization for the TSP. Let $\mathcal{C}(d, n)$ denote the collection of all d-dimensional $n \times n \times \cdots \times n$ cost arrays C for which all feasible solutions of the $(d, d-1)$-AP have the same objective value. Clearly $\mathcal{C}(d, n)$ is a linear subspace of the set of all d-dimensional $n \times n \times \cdots \times n$ arrays. Our goal is to prove that

$$\mathcal{C}(d, 3) = \mathrm{SAVS}(d, d-1, 3). \tag{5.9}$$

To that end, we consider all feasible solutions of the $(d, d-1)$-AP for $n = 3$. Next we build up the 0-1 matrix M_d where the rows of M_d correspond to the feasible solutions and the columns correspond to the d-tuples over $\{1, 2, 3\}$. The entry of M_d that corresponds to the feasible solution F and the d-tuple $(i_1, i_2, \dots, i_d)$ is set to 1 if and only if $(i_1, i_2, \dots, i_d) \in F$.

Note that every row of the matrix M_{d+1} is obtained from three disjoint rows of the matrix M_d. For every row r_1 of the matrix M_d there are exactly two rows r_2, r_3 disjoint with r_1, and r_2 and r_3 are also mutually disjoint. Therefore $r_1 r_2 r_3$ and $r_1 r_3 r_2$ are rows of M_{d+1}. Hence the matrix M_{d+1} has twice as many rows as M_d. This corresponds to the fact that for $n = 3$ the number of feasible solutions doubles when moving from the planar d-dimensional assignment problem to the $(d+1)$-dimensional one. It is easy to see that M_d is a $3 \cdot 2^{d-1} \times 3^d$ matrix. The following matrices M_d, $d = 1, 2$ are provided as illustration:

$$M_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad M_2 = \left(\begin{array}{ccc|ccc|ccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \end{array}\right).$$

$\mathcal{C}(d, 3)$ is the solution space of the system of linear equations with coefficient matrix M_d and a constant right hand side vector. Thus we obtain

$$\dim \mathcal{C}(d,3) = 3^d + 1 - \operatorname{rank}(M_d).$$

From Proposition 5.6 (iv) we know that $\dim(\mathrm{SAVS}(d, d-1, 3)) = 3^d - 2^d$. Hence in order to prove that (5.9) holds, we need to show that $\operatorname{rank}(M_d) = 2^d + 1$. Observe that in fact it suffices to show that $\operatorname{rank}(M_d) \geq 2^d + 1$ since obviously $\mathrm{SAVS}(d, d-1, 3) \subseteq \mathcal{C}(d,3)$; Lemma 5.12 below completes the proof. □

Lemma 5.12. *Let M_d be the matrix constructed above. We have*

$$\operatorname{rank}(M_d) \geq 2^d + 1.$$

Proof. We start with observing the following recursive structure of M_d. Define

$$A_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad B_0 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \qquad C_0 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix},$$

and recursively for $k \geq 1$

$$A_k = \begin{pmatrix} A_{k-1} & B_{k-1} & C_{k-1} \\ A_{k-1} & C_{k-1} & B_{k-1} \end{pmatrix} \qquad B_k = \begin{pmatrix} B_{k-1} & C_{k-1} & A_{k-1} \\ B_{k-1} & A_{k-1} & C_{k-1} \end{pmatrix}$$

$$C_k = \begin{pmatrix} C_{k-1} & A_{k-1} & B_{k-1} \\ C_{k-1} & B_{k-1} & A_{k-1} \end{pmatrix},$$

where A_k, B_k and C_k are $3 \cdot 2^k \times 3^{k+1}$ matrices. It is easy to see that $M_d = A_{d+1}$ for $d \geq 1$. Next we will exhibit a regular $(2^d+1) \times (2^d+1)$ submatrix M'_d of M_d which will settle the lemma. We construct new matrices A'_k, B'_k and C'_k from A_k, B_k and C_k as follows: First, remove all columns with indices $\geq 2 \cdot 3^k + 1$. Next, remove all rows and columns with indices that are divisible by 3. It is straightforward to observe that the recursive structure survives this construction. More precisely we have

$$A'_k = \begin{pmatrix} A'_{k-1} & B'_{k-1} \\ A'_{k-1} & C'_{k-1} \end{pmatrix} \quad B'_k = \begin{pmatrix} B'_{k-1} & C'_{k-1} \\ B'_{k-1} & A'_{k-1} \end{pmatrix} \quad C'_k = \begin{pmatrix} C'_{k-1} & A'_{k-1} \\ C'_{k-1} & B'_{k-1} \end{pmatrix} \tag{5.10}$$

for $k \geq 1$ and

$$A'_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad B'_0 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad C'_0 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

The matrices A'_k, B'_k and C'_k have 2^{k+1} rows and 2^{k+1} columns. We obtain our target matrix M'_d from the matrix A'_{d+1} by re-inserting row 3 and column 3 of the matrix A_{d+1}. In order to show that M'_d is regular, we will calculate its determinant by a recursive approach. We will make use of the observation that the upper left and lower left block are identical in the matrices A'_k, B'_k and C'_k. This will allow us to create a zero block as lower left block of a reduced matrix which has the same determinant as A'_k. This results in

$$\det A'_k = \det A'_{k-1} \det \left(C'_{k-1} - B'_{k-1}\right) \tag{5.11}$$

for $k \geq 1$. An analogous argument yields

$$\det (C'_k - B'_k) = \det \left(C'_{k-1} - B'_{k-1}\right) \det \left(B'_{k-1} + C'_{k-1} - 2A'_{k-1}\right) \tag{5.12}$$

and

$$\det (B'_k + C'_k - 2A'_k) = \det \left(B'_{k-1} + C'_{k-1} - 2A'_{k-1}\right) \det \left(3\left(B'_{k-1} - C'_{k-1}\right)\right) \tag{5.13}$$

for $k \geq 1$. Furthermore observe that

$$\det \left(3\left(B'_{k-1} - C'_{k-1}\right)\right) = 3^{2^k} \det \left(C'_{k-1} - B'_{k-1}\right) \tag{5.14}$$

as the involved matrices are of size $2^k \times 2^k$. Let

$$z_k = \det A'_k, \; u_k = \det (C'_k - B'_k), \; v_k = \det (B'_k + C'_k - 2A'_k).$$

By explicit calculations we get the initial values $z_0 = 1, u_0 = -1, v_0 = 3$.

From (5.11)–(5.14) we obtain the following recursions for $k \geq 1$

$$z_k = z_{k-1}u_{k-1}, \qquad u_k = u_{k-1}v_{k-1}, \qquad v_k = 3^{2^k} v_{k-1}u_{k-1}. \tag{5.15}$$

By combining the second and the third equation in (5.15) we obtain $v_k = 3^{2^k} u_k$ which allows to eliminate v_k. We obtain the new system of recursions

$$z_k = z_{k-1}u_{k-1}, \qquad u_k = 3^{2^{k-1}} u_{k-1}^2, \qquad k \geq 1. \tag{5.16}$$

This already implies that all matrices A'_k are regular, but for the sake of completeness we provide the solution for the recursion above. It is not hard to show that

$$u_k = 3^{k2^{k-1}}, \quad z_k = 3^{(k-2)2^{k-1}+1}$$

provides a solution to the system (5.16) with the initial conditions $z_0 = 1$ and $u_0 = -1$. As a consequence thereof we get that

$$\det A'_{d+1} = 3^{d2^{d+1}+1}.$$

Note that M'_d differs from A'_{d+1} only in its additional row and additional column. The additional column (the third column) of M'_d corresponds to the third unit vector. By developing the determinant of M'_d with respect to this column, we obtain

$$\det M'_d = \det A'_{d+1} = 3^{d2^{d+1}+1},$$

which implies that M'_d is regular and hence $\operatorname{rank}(M_d) \geq 2^d + 1$. □

Let us mention that one can show that $\operatorname{rank}(M_d) = 2^d + 1$ by calculating the reduced row echelon form of matrix M_d. For our purposes it sufficed to show a weaker result which could be obtained more elegantly.

5.2.4 The COVP for the general case: (d, s)-AP

Theorem 5.7 and Theorem 5.11 impose the following question for the (d, s)-AP.

Question 5.13. *Is it true that a feasible instance of the (d, s)-AP with cost* *array C has the COVP if and only if C is a sum-decomposable array with parameters d and s?*

Theorem 5.7 and Theorem 5.11 imply that the answer to Question 5.13 is affirmative in the following cases: $(2,1)$-AP, $(3,1)$-AP, $(3,2)$-AP, $(4,1)$-AP and $(4,3)$-AP. This leaves us with the $(4,2)$-AP as the smallest unsettled case. This is also the smallest case for which it is not guaranteed that a feasible solution exists for all $n \geq 2$.

The following example shows that the answer to Question 5.13 is negative in general.

Example 5.14. There are 72 Graeco-Latin squares of size 3, hence there are 72 feasible solutions for the $(4,2)$-AP with $n = 3$, see Observation 2.8. We consider the system of linear equations that is obtained by requiring that all 72 feasible solutions have the same objective value. The dimension of the solution space of this system of equations, and thus the dimension of the space of cost arrays with the COVP, is 49, which can easily be calculated by a computer algebra system. By Proposition 5.6 one gets that the dimension of SAVS(4,2,3) is 33. Hence, there exists a cost array with the COVP that is not sum-decomposable. Now we provide one such array.

Let C be the $3 \times 3 \times 3 \times 3$ array where $c(1,1,1,2)$, $c(1,1,2,1)$, $c(1,2,1,1)$, $c(1,2,2,2)$, $c(2,1,1,1)$, $c(2,1,2,2)$, $c(2,2,1,2)$, $c(2,2,2,1)$ and $c(3,3,3,3)$ have value 1 and all other entries have value 0. All 72 feasible solutions of the (4,2)-AP with this cost array have the objective value 1, and it is easy to check that C is not sum-decomposable.

We did not find counterexamples for the $(4,2)$-AP for $n \geq 4$. For $n = 4$ and $n = 5$ the computer calculations gave the affirmative answer to Question 5.13. For $n = 6$ there are no feasible solutions and for $n = 7$ the number of feasible solutions gets too large to handle.

Conjecture 5.15. *A feasible instance of the $(4,2)$-AP with cost array C of size $n \neq 3$ has the COVP if and only if C is a sum-decomposable array with parameters 4 and 2.*

We believe that Example 5.14 occurs since for $n = 3$ the number of feasible solutions is relatively small, but then grows very fast. Note that for larger values of n even the number of Graeco-Latin squares is unknown.

This eliminates explicit proof approaches as the set of feasible solutions is not known. A proof would need to exploit the structure of the set of Graeco-Latin squares.

We checked that the answer to Question 5.13 for the $(5, 2)$-AP for $n = 4$ is affirmative. For $n = 2, 3, 6$ there are no feasible solutions. For $n = 5$ the set of feasible solutions became too large for our straightforward computational approach. The same happened for the $(5, 3)$-AP for $n = 4$, and for $n = 2, 3$ the problem is again infeasible. The motivation behind our experiments was our wish to obtain a feeling whether the answer to Question 5.13 is affirmative for sufficiently large n. We believe so, but we could handle only very small cases and do not have enough empirical results to propose a conjecture.

5.3 The COVP for d-dimensional transportation problems

In this section we deal with the COVP for d-dimensional transportation problems. Specifically, we show that our COVP characterization for the axial d-dimensional assignment problem carries over to the axial d-dimensional transportation problem while this approach fails for the more involved planar case.

Multidimensional transportation problems are known in the literature under diverse names. Alternative names are for example multi-index or d-index transportation problems, d-fold transportation problems and multi-way or d-way transportation problems, see e.g. [31, 72].

The d-dimensional transportation problem can be defined along the lines of the definition of the d-dimensional assignment problem (d, s)-AP. We are given a d-dimensional $n_1 \times n_2 \times \cdots \times n_d$ cost array C. While in the assignment case the right hand side of all equality constraints is equal to one, in the transportation case we are additionally given an s-dimensional array B^Q for each set $Q \in \mathcal{Q}_s$ of fixed indices which provides the right hand side values for this group of constraints induced by the set Q. The arrays B^Q can be viewed as marginals for the transportation array $X = (x(i_1, i_2, \ldots, i_d))$. We refer to the resulting transportation problem as (d, s)-

TP.

Like for the assignment case, we obtain the *axial d-dimensional transportation problem* when $s = 1$ and the *planar d-dimensional transportation problem* when $s = d - 1$. As we will deal with the axial d-dimensional transportation problem below, we provide its explicit formulation.

We are given an $n_1 \times n_2 \times \cdots \times n_d$ cost array $C = (c(i_1, i_2, \ldots, i_d))$ and d supply-demand vectors $B_1, \ldots, B_d$, where the k-th vector $B_k = (b_k(i))$ is an n_k-dimensional vector over the nonnegative integers. Furthermore we assume $\sum_{i=1}^{n_1} b_1(i) = \sum_{i=1}^{n_2} b_2(i) = \cdots = \sum_{i=1}^{n_d} b_d(i)$. Let $I_r = \{1, \ldots, n_r\}$ be the index set for i_r, $r = 1, \ldots, d$. We obtain the following formulation for the $(d, 1)$-TP:

$$\begin{aligned}
\min & \sum_{i_1 \in I_1} \sum_{i_2 \in I_2} \cdots \sum_{i_d \in I_d} c(i_1, i_2, \ldots, i_d) x(i_1, i_2, \ldots, i_d) \\
\text{s.t.} & \sum_{\substack{i_1 \in I_1, \ldots, i_d \in I_d \\ \text{s.t. } i_k = j}} x(i_1, i_2, \ldots, i_d) = b_k(j) \quad \text{for all } k \in \{1, \ldots, d\}, j \in \{1, \ldots, n_k\} \\
& x(i_1, i_2, \ldots, i_d) \geq 0 \qquad\qquad \text{for all } i_r = 1, \ldots, n_r,\ r = 1, \ldots, d.
\end{aligned}$$

If $X = (x(i_1, \ldots, i_n))$ has to be integral, the problem above becomes NP-hard for $d \geq 3$. For $d = 2$ the well-known classical Hitchcock transportation problem arises.

Theorem 5.16. *An instance of the axial d-dimensional transportation problem with cost array C has the COVP if and only if C is sum-decomposable array with parameters d and 1.*

Proof. Any instance of the integral axial d-dimensional transportation problem can be transformed into an equivalent instance of the axial d-dimensional assignment problem. To that end, we replace every supply/demand facility that has a supply/demand value $t > 1$ by t facilities with identical transportation costs that have supply/demand value 1. In this manner we get an equivalent problem with a blown up $n \times n \times \cdots \times n$ cost array where $n = \sum_{i=1}^{n_1} b_1(i)$ and all supplies/demands are 1. Thus the newly obtained problem is the $(d, 1)$-AP.

For the integral version of the $(d, 1)$-TP we can apply the COVP characterization from Theorem 5.7 directly. For the non-integral version observe that the transformed problem with unit supplies and demands is a relaxation of the $(d, 1)$-AP which results if the integrality constraints on X are dropped. In this case it follows from Theorem 5.7 that the set of instances with the COVP is a subspace of $\mathrm{SAVS}(d, 1, n)$, and is hence equal to $\mathrm{SAVS}(d, 1, n)$. Note that the transformation that blows up the cost array and the inverse transformation preserve the sum-decomposability property of the cost array. $\square$

Note that setting $d = 2$ in Theorem 5.16 implies that an instance of the classical transportation problem with cost matrix C has the COVP if and only if C is a sum matrix. The proof of Theorem 5.16 provides the connection to assignment problems and further to Berenguer's COVP characterization for the TSP, cf. Theorem 5.3. As a by-product this reveals the nature of the connection between results of Klinz and Woeginger [54] on the optimality of the North-West corner rule and Theorem 5.16, and thus answers an open problem mentioned in the concluding section of [54].

At first sight one might expect that Theorem 5.11 for the planar d-dimensional assignment problem $(d, d-1)$-AP carries over to the planar d-dimensional transportation problem $(d, d-1)$-TP. However several difficulties arise in this case. First, note that the blow-up technique to transform the transportation problem to a (continuous) assignment problem does not work in general in the planar setting. The second and probably bigger obstacle to a COVP characterization for the planar case comes from the fact that for $d \geq 3$ the d-dimensional planar transportation problem does not necessarily have feasible solutions (not even in the non-integral case). Due to the universality result of de Loera and Onn [31] checking feasibility for the 3-dimensional planar (integer) transportation problem is as hard as deciding whether a general linear (integer) program has a feasible solution (the result already holds for a fixed third dimension, i.e., for $n_3 = 3$). As the number of feasible solutions of a feasible instance of the d-dimensional transportation problem can be as small as one, even in the non-integral case, it is not any longer necessary for the COVP that all dual constraints have to be fulfilled with equality. Hence the approach based on the complementarity slackness condition that works for the linear assignment problem and the classical transportation problem, that was explained in the introduction, fails for $d \geq 3$.

Concluding, there does not seem to be much hope to be able to provide a nice sufficient and necessary condition for the set of instances with the COVP for the 3-dimensional planar transportation problem and even less hope for cases with $d > 3$.

5.4 The COVP for spanning tree, shortest path and matching problems

In this section we provide COVP characterization for the minimum spanning tree problem, the shortest path problem and the minimum weight maximum cardinality matching problem.

5.4.1 The COVP for the minimum spanning tree problem

In the minimum spanning tree problem (MST) we are given a connected, undirected graph $G = (V, E)$ and edge weights w_e for each edge $e \in E$. The task is to find a spanning tree for which the sum of edge weights is minimal.

Lemma 5.17. *Let I be an instance of the MST with graph G and weights $w = (w_e)$. If I has the COVP, then every edge in any (simple) cycle in G has the same weight.*

Proof. Let C be a (simple) cycle in G and e be an arbitrary edge from C. There exists a spanning tree T which contains all edges of C except e. By adding edge e to T and removing from T in turn an arbitrary edge $f \neq e$ from C, we obtain another spanning tree T'. As the weights of T and T' are identical, it follows that $w_e = w_f$. Hence all edges in C have the same weight. $\square$

To formulate the COVP characterization for the MST we need the following definition.

Definition 5.18. Let $G = (V, E)$ be an undirected graph. The undirected graph $H = (V_H, E_H)$ which has a vertex v_e for each edge $e \in E$ and an edge $\{v_e, v_f\} \in E_H$ if and only if e and f lie on a common simple cycle C is called *cycle graph* of G.

Theorem 5.19. *Let I be an instance of the MST problem with graph G and weights $w = (w_e)$. Let H be the cycle graph of G and let $V_1, \dots, V_\ell$ be the vertex sets of connected components of H and $E_1, \dots, E_\ell$ be the corresponding sets of edges in G. Then I has the COVP if and only if there exist constants α_i, $i = 1, \dots, \ell$ such that for all $e \in E_i$ $w_e = \alpha_i$, for all i.*

Proof. To prove that the stated condition is sufficient, let T and T' be two spanning trees. It is easy to see that one can move from T to T' by a sequence of moves which add an edge $e \in T' \setminus T$ and delete an edge $f \in T \setminus T'$ where f lies on the unique cycle in $T \cup \{e\}$. As all edges on the cycle have the same weight, it follows from iterative application that $w(T) = w(T')$, where $w(T)$ denotes the sum of all w_e, $e \in T$.

To prove necessity of the stated condition, first observe that every bridge of G (corresponds to an isolated vertex in H) is part of every spanning tree and hence can have arbitrary weight. The claim now follows by applying Lemma 5.17 for each cycle C in G. □

Note that if H is connected (which is the case for example if G is 2-connected), then the COVP holds if all edges have the same weight.

It is easy to see that the COVP characterization for the MST problem can be carried over to the setting of matroids (circuits play the role of cycles and bases play the role of spanning trees).

5.4.2 The COVP for the shortest path problem

Given a weighted graph (undirected or directed) with the vertex set $V = \{1, 2, ..., n\}$ the shortest path problem is the problem of finding a path from vertex 1 to vertex n such that the sum of edge weights along the path is minimized. In what follows we consider both the undirected and the directed version of the shortest path problem in a complete graph and provide COVP characterizations.

Theorem 5.20. *Let $G = (V, E)$ be the complete undirected graph with the vertex set $V = \{1, 2, \dots, n\}$, $n \geq 3$, and let $w(i, j)$ denote the nonnegative weight of the edge (i, j). This instance of the undirected shortest path problem has the COVP if and only if the weights are of the following form*

$$w(i,j) = w(j,i) = \begin{cases} a & \text{if } i = 1, j \neq n, \\ b & \text{if } i \neq 1, j = n, \\ a + b & \text{if } i = 1, j = n, \\ 0 & \text{otherwise} \end{cases} \tag{5.17}$$

for some non-negative reals a and b.

Proof. Assume that every path from 1 to n has the same weight. For $n = 3$ the result is straightforward. Assume $n \geq 4$ and take two distinct vertices i and j such that $1 < i, j < n$. Consider the five paths from vertex 1 to vertex n that only go through a subset of the vertices $\{1, i, j, n\}$. By assumption we get the following relations

$$\begin{aligned} w(1,n) &= w(1,i) + w(i,j) + w(j,n) \\ &= w(1,j) + w(j,i) + w(i,n) \\ &= w(1,i) + w(i,n) \\ &= w(1,j) + w(j,n). \end{aligned}$$

By adding and subtracting appropriate equations we get that $w(i, j) = 0$, $w(1, i) = w(1, j)$, $w(i, n) = w(j, n)$, so (5.17) follows.

Note that the converse trivially holds, which concludes the proof. □

Theorem 5.21. *Let $G = (V, E)$ be the complete directed acyclic graph with the vertex set $V = \{1, 2, \dots, n\}$ and edge set $E = \{(i, j) \in V \times V : i < j\}$, and let $w(i, j)$ denote the weight of edge (i, j). This instance of the directed shortest path problem has the COVP if and only if there exists a real vector $A = (a_i)$ such that*

$$w(i,j) = a_j - a_i \qquad \text{for all } \ i, j \in \{1, \dots, n\}, \ i < j. \tag{5.18}$$

Proof. Assume that every path from vertex 1 to vertex n has the same weight. Consider the path composed of edges $(1, i)$, (i, j) and (j, n). It has the same weight as the path composed of edges $(1, j)$ and (j, n). It follows that $w(i, j) = w(1, j) - w(1, i)$. Set $a_i := w(1, i)$ for $i = 1, \dots, n$, so that $w(i, j) = a_j - a_i$.

Now assume that for all $i < j$ the weight of (i, j) can be represented as in (5.18) for some vector $A = (a_i)$. Consider an arbitrary path from vertex 1 to n, and let $1 = v_1 < v_2 < \dots < v_k = n$ be all vertices on that path. Then the weight of the path is

$$\sum_{i=1}^{k-1} w(v_i, v_{i+1}) = \sum_{i=1}^{k-1} a_{v_{i+1}} - a_{v_i} = a_{v_k} - a_{v_1} = a_n - a_1.$$

Since this number is independent of the choice of path, (5.18) is also sufficient and hence the statement holds. □

5.4.3 The COVP for the minimum weight maximum cardinality matching problem

In the minimum weight maximum cardinality matching problem we are given an undirected graph $G = (V, E)$ and edge weights $w(i, j)$ for each edge $(i, j) \in E$. Our goal is to find a matching for which the sum of edge weights is minimal among all matchings of maximal cardinality.

Theorem 5.22. *Let I be an instance of the minimum weight maximum cardinality matching on the complete undirected graph G with n vertices and edge weights $w(i, j)$.*

(*i*) *If n is odd, I has the COVP if and only if all edge weights are equal.*

(*ii*) *If n is even, I has the COVP if and only if there exists a real vector $A = (a_i)$ such that*

$$w(i, j) = a_i + a_j \qquad \text{for all } i \neq j. \tag{5.19}$$

Proof. Let n be odd. Suppose that every maximum cardinality matching has the same weight. Let $i, j, k \in V$ be three distinct vertices. Let M be a maximum cardinality matching on the vertex set $V \setminus \{i, j, k\}$. By adding an arbitrary edge from the triangle defined by i, j and k to M we obtain a maximum cardinality matching on the initial instance. By assumption it follows that every edge in the triangle defined by i, j and k has the same weight. Hence the statement follows.

Let n be even. Assume that every perfect matching has the same weight. Since each of the two pairs of edges (i, j), (k, l) and (i, l), (j, k) can be identically extended to a perfect matching it follows that

$$w(i, j) + w(k, l) = w(i, l) + w(j, k)$$

for all distinct i, j, k, l. Hence, there exist two real vectors $U = (u_i)$ and $V = (v_i)$ such that $w(i, j) = u_i + v_j$ for all $i \neq j$. Since the weight matrix has to be symmetric (G is undirected), there exists a real vector $A = (a_i)$ such that (5.19) holds.

Note that (5.19) is clearly a sufficient condition for the COVP. □

5.5 Conclusions and open problems

In this chapter our goal was to characterize the set of instances with the constant objective value property (COVP), i.e. to investigate the space of all instances for which every feasible solution has the same objective value.

As our central result, we showed that the COVP instances of the planar and the axial d-dimensional assignment problem are characterized by sum-decomposable arrays with the corresponding parameters. We provided a counterexample which shows that these results do not carry over to general d-dimensional assignment problem. The following remains a challenging open problem.

Open Problem 5.23. Determine whether the sum decomposability characterizes the COVP instances in all cases of the multidimensional assignment problems for which feasible solutions exist and size of the instance is sufficiently large.

We used the results for the axial d-dimensional assignment problem to characterize the COVP instances for the axial d-dimensional transportation problem.

Furthermore, as simpler side results, we characterized the COVP instances for the following classical combinatorial optimization problems: the minimum spanning tree, the shortest path problem in undirected and directed graphs and the minimum weight cardinality matching problem in complete graphs.

Open Problem 5.24. Find a combinatorial optimization problem with a sum objective value function for which a nice COVP characterization can be obtained where the underlying cost structure is not essentially sum-decomposable.

Bibliography

[1] A. Aggarwal and J.K. Park, *Sequential searching in multidimensional monotone arrays*, Technical Report RC 15128, IBM T.J. Watson Research Center, Yorktown Height, New York, November 1989.

[2] ———, *Improved algorithms for economic lot size problems*, Oper. Res. **41** (1993), 549–571.

[3] M. Akgül, *A genuinely polynomial primal simplex algorithm for the assignment problem*, Discrete Appl. Math. **45** (1993), 93–115.

[4] G. Appa, D. Magos, and I. Mourtos, *A branch and cut algorithm for a four-index assignment problem*, J. Oper. Res. Soc. **55** (2004), 298–307.

[5] ———, *A new class of facets for the Latin square polytope*, Discrete Appl. Math. **154** (2006), 900–911.

[6] ———, *On multi-index assignment polytopes*, Linear Algebra Appl. **416** (2006), 224–241.

[7] C. Arbib, D. Pacciarelli, and S. Smriglio, *A three-dimensional matching model for perishable production scheduling*, Discrete Appl. Math. **92** (1999), 1–15.

[8] E.M. Arkin and R. Hassin, *On local search for weighted k-set packing*, Math. Oper. Res. **23** (1998), 640–648.

[9] E. Balas and P.R. Landweer, *Traffic assignment in communication satellites*, Oper. Res. Lett. **2** (1983), 141–147.

[10] E. Balas and M.J. Saltzman, *An algorithm for the three-index assignment problem*, Oper. Res. **39** (1991), 150–161.

[11] M.L. Balinski and R.E. Gomory, *A primal method for the assignment and transportation problems*, Management Science **10** (1964), 578–593.

[12] A. Barvinok, S.P. Fekete, D.S. Johnson, A. Tamir, G.J. Woeginger, and R. Woodroofe, *The geometric maximum traveling salesman problem*, J. ACM **50** (2003), 641–664.

[13] A.I. Barvinok, *Two algorithmic results for the traveling salesman problem*, Math. Oper. Res. **21** (1996), 65–84.

[14] X. Berenguer, *A characterization of linear admissible transformations for the m-travelling salesmen problem*, European J. Oper. Res. **3** (1979), 232–238.

[15] D.P. Bertsekas, *A new algorithm for the assignment problem*, Math. Programming **21** (1981), 152–171.

[16] G. Birkhoff, *Three observations on linear algebra*, Univ. Nac. Tucumán. Revista A. **5** (1946), 147–151.

[17] R.C. Bose, S.S. Shrikhande, and E.T. Parker, *Further results on the construction of mutually orthogonal Latin squares and the falsity of Euler's conjecture*, Canad. J. Math. **12** (1960), 189–203.

[18] D. Briskorn, A. Drexl, and F.C.R. Spieksma, *Round robin tournaments and three index assignments*, 4OR **8** (2010), 365–374.

[19] R.E. Burkard, *Admissible transformations and assignment problems*, Vietnam J. Math. **35** (2007), 373–386.

[20] R.E. Burkard, V.G. Deineko, R. van Dal, J.A.A. van der Veen, and G.J. Woeginger, *Well-solvable special cases of the traveling salesman problem: A survey*, SIAM Rev. **40** (1998), 496–546.

[21] R.E. Burkard, M. Dell'Amico, and S. Martello, *Assignment Problems*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[22] R.E. Burkard, B. Klinz, and R. Rudolf, *Perspectives of Monge properties in optimization*, Discrete Appl. Math. **70** (1996), 95–161.

[23] R.E. Burkard, R. Rudolf, and G.J. Woeginger, *Three-dimensional axial assignment problems with decomposable cost coefficients*, Discrete Appl. Math. **65** (1996), 123–139.

[24] R.E. Burkard and U. Zimmermann, *Combinatorial optimization in linearly ordered semimodules: a survey*, Modern applied mathematics (Bonn, 1979), North-Holland, Amsterdam, 1982, pp. 391–436.

[25] C.J. Colbourn and J.H. Dinitz, *Handbook of combinatorial designs, second edition (discrete mathematics and its applications)*, Chapman & Hall/CRC, 2006.

[26] Y. Crama, A. Oerlemans, and F.C.R. Spieksma, *Production planning in automated manufacturing*, Lecture Notes in Economics and Mathematical Systems, vol. 414, Springer-Verlag, Berlin, 1994.

[27] Y. Crama and F.C.R. Spieksma, *Approximation algorithms for three-dimensional assignment problems with triangle inequalities*, European J. Oper. Res. **60** (1992), 273–279.

[28] A. Ćustić and B. Klinz, *The constant objective value property for combinatorial optimization problems*, submitted, arXiv:1405.6096.

[29] A. Ćustić, B. Klinz, and G.J. Woeginger, *Geometric versions of the three-dimensional assignment problem under arbitrary norms*, in preparation.

[30] ______, *Planar 3-dimensional assignment problems with Monge-like cost arrays*, in preparation, arXiv:145.5210.

[31] J.A. De Loera and S. Onn, *All linear and integer programs are slim 3-way transportation programs*, SIAM J. Optim. **17** (2006), 806–821 (electronic).

[32] U. Derigs and U. Zimmermann, *An augmenting path method for solving linear bottleneck assignment problems*, Computing **19** (1977/78), 285–295.

[33] E.A. Dinic and M.A. Kronrod, *An algorithm for solving the assignment problem*, Dokl. Akad. Nauk SSSR **189** (1969), 23–25.

[34] M.E. Dyer and A.M. Frieze, *Planar* 3*DM is NP-complete*, J. Algorithms **7** (1986), 174–184.

[35] R. Euler, R.E. Burkard, and R. Grommes, *On Latin squares and the facial structure of related polytopes*, Discrete Math. **62** (1986), 155–181.

[36] R. Euler and H. Le Verge, *Time-tables, polyhedra and the greedy algorithm*, Discrete Appl. Math. **65** (1996), 207–221.

[37] D.G. Fon-Der-Flaass, *Arrays of distinct representatives—a very simple NP-complete problem*, Discrete Math. **171** (1997), 295–298.

[38] D. Fortin and A. Tusera, *Routing in meshes using linear assignment*, Operations Research '93, 1994, pp. 169–171.

[39] A.M. Frieze, *Complexity of a* 3*-dimensional assignment problem*, European J. Oper. Res. **13** (1983), 161–164.

[40] A.M. Frieze and G.B. Sorkin, *Efficient algorithms for three-dimensional axial and planar random assignment problems*, Random Struct. Alg. (2014), in print.

[41] M.R. Garey and D.S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*, W. H. Freeman & Co., New York, NY, USA, 1979.

[42] K.C. Gilbert and R.B. Hofstra, *An algorithm for a class of three-dimensional assignment problems arising in scheduling applications*, IIE Transactions **19** (1987), 29–33.

[43] P.C. Gilmore, E.L. Lawler, and D.B. Shmoys, *Well-solved special cases*, The traveling salesman problem, Wiley-Intersci. Ser. Discrete Math., Wiley, Chichester, 1985, pp. 87–143.

[44] È.Kh. Gimadi and Yu.V. Glazkov, *An asymptotically exact algorithm for solving a modified three-index planar assignment problem*, Diskretn. Anal. Issled. Oper. Ser. 2 **13** (2006), 10–26.

[45] A.V. Goldberg and R. Kennedy, *An efficient cost scaling algorithm for the assignment problem*, Math. Programming **71** (1995), 153–177.

[46] M. Hall, *An existence theorem for Latin squares*, Bull. Amer. Math. Soc. **51** (1945), 387–388.

[47] P. Hansen and L. Kaufman, *A primal-dual algorithm for the three-dimensional assignment problem*, Cahiers Centre Études Recherche Opér. **15** (1973), 327–336.

[48] A.J.W. Hilton, *The reconstruction of Latin squares with applications to school timetabling and to experimental design*, Math. Programming Stud. (1980), 68–77.

[49] C.G.J. Jacobi, *De investigando ordine systematis aequationum differentialum vulgarium cujuscunque*, Borchardt Journal für die reine und angewandte Mathematik **LXIV** (1865), 297–320.

[50] Vo-Khac K., *La régularisation dans les problèmes combinatoires et son application au problème de sectorisation*, Rev. Française Informat. Recherche Opérationnelle **5** (1971), 59–77.

[51] M.-Y. Kao, T.-W. Lam, W.-K. Sung, and H.-F. Ting, *A decomposition theorem for maximum weight bipartite matchings*, SIAM J. Comput. **31** (2001), 18–26.

[52] R.M. Karp, *Reducibility among combinatorial problems*, Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972), Plenum, New York, 1972, pp. 85–103.

[53] B. Klinz and G.J. Woeginger, *A new efficiently solvable special case of the three-dimensional axial bottleneck assignment problem*, Combinatorics and computer science (Brest, 1995), Lecture Notes in Comput. Sci., vol. 1120, Springer, Berlin, 1996, pp. 150–162.

[54] ______, *The northwest corner rule revisited*, Discrete Appl. Math. **159** (2011), 1284–1289.

[55] H.W. Kuhn, *The Hungarian method for the assignment problem*, Naval Res. Logist. Quart. **2** (1955), 83–97.

[56] H.W. Lenstra, *Integer programming with a fixed number of variables*, Math. Oper. Res. **8** (1983), 538–548.

[57] J.K. Lenstra and A.H.G. Rinnooy Kan, *A characterization of linear admissible transformations for the m-travelling salesmen problem: A result of Berenguer*, European J. Oper. Res. **3** (1979), 250–252.

[58] D. Magos, *Tabu search for the planar three-index assignment problem*, J. Global Optim. **8** (1996), 35–48.

[59] D. Magos and P. Miliotis, *An algorithm for the planar three-index assignment problem*, European J. Oper. Res. **77** (1994), 141–153.

[60] R.A. Murphey, P.M. Pardalos, and L. Pitsoulis, *A parallel grasp for the data association multidimensional assignment problem*, Parallel processing of discrete problems (Minneapolis, MN, 1997), IMA Vol. Math. Appl., vol. 106, Springer, New York, 1999, pp. 159–179.

[61] K.R. Pattipati, S. Deb, Y. Bar-Shalom, and R.B. Washburn, *A new relaxation algorithm and passive sensor data association*, Automatic Control, IEEE Transactions on **37** (1992), 198–213.

[62] U. Pferschy, R. Rudolf, and G.J. Woeginger, *Some geometric clustering problems*, Nordic J. Comput. **1** (1994), 246–263.

[63] W.P. Pierskalla, *The tri-substitution method for the three-dimensional assignment problem*, Journal of the Canadian Operations Research Society **5** (1967), 71–81.

[64] ______, *The Multidimensional Assignment Problem*, Oper. Res. **16** (1968), 422–431.

[65] A. Polishchuk and L. Positselski, *Quadratic algebras*, University Lecture Series, vol. 37, American Mathematical Society, Providence, RI, 2005.

[66] S. Polyakovskiy, F.C. Spieksma, and G.J. Woeginger, *The three-dimensional matching problem in Kalmanson matrices*, J. Comb. Optim. **26** (2013), 1–9.

[67] A.B. Poore, *Multidimensional assignment formulation of data association problems arising from multitarget and multisensor tracking*, Comput. Optim. Appl. **3** (1994), 27–57.

[68] A.B. Poore and N. Rijavec, *A Lagrangian relaxation algorithm for multidimensional assignment problems arising from multitarget tracking*, SIAM J. Optim. **3** (1993), 544–563.

[69] J. Pusztaszeri, *The nonlinear assignment problem in experimental high energy physics*, Nonlinear assignment problems, Comb. Optim., vol. 7, Kluwer Acad. Publ., Dordrecht, 2000, pp. 55–89.

[70] J. Pusztaszeri, P.E. Rensing, and T.M. Liebling, *Tracking elementary particles near their primary vertex: a combinatorial approach*, J. Global Optim. **9** (1996), 41–64.

[71] L. Qi and D. Sun, *Polyhedral methods for solving three index assignment problems*, Nonlinear assignment problems, Comb. Optim., vol. 7, Kluwer Acad. Publ., Dordrecht, 2000, pp. 91–107.

[72] M. Queyranne and F.C.R Spieksma, *Multi-index transportation problems, in C. A. Floudas and P. M. Pardalos, Encyclopedia of Optimization*, Springer, 2009.

[73] H.J. Ryser, *A combinatorial theorem with an application to latin rectangles.*, Proc. Am. Math. Soc. **2** (1951), 550–552.

[74] E.D. Schell, *Distribution of a product by several properties*, Proceedings of the Second Symposium in Linear Programming, Washington, D. C., 1955, National Bureau of Standards, Washington, D. C., 1955, pp. 615–642.

[75] W. Schnyder, *Embedding planar graphs on the grid*, Proc. 1-st Annual ACM-SIAM Symp. On Discr. Alg. (SODA) (1990), 138–147.

[76] F.C.R. Spieksma, *Multi index assignment problems: complexity, approximation, applications*, Nonlinear assignment problems, Comb. Optim., vol. 7, Kluwer Acad. Publ., Dordrecht, 2000, pp. 1–12.

[77] F.C.R. Spieksma and G.J. Woeginger, *Geometric three-dimensional assignment problems*, European J. Oper. Res. **91** (1996), 661–618.

[78] J.M.M. van Rooij, M.E. van Kooten Niekerk, and H.L. Bodlaender, *Partition into triangles on bounded degree graphs*, Theory Comput. Syst. **52** (2013), 687–718.

[79] M. Vlach, *Branch and bound method for the three-index assignment problem*, Ekonom.-Mat. Obzor **3** (1967), 181–191.

[80] D. Yokoya, C.W. Duin, and T. Yamada, *A reduction approach to the repeated assignment problem*, European J. Oper. Res. **210** (2011), 185–193.