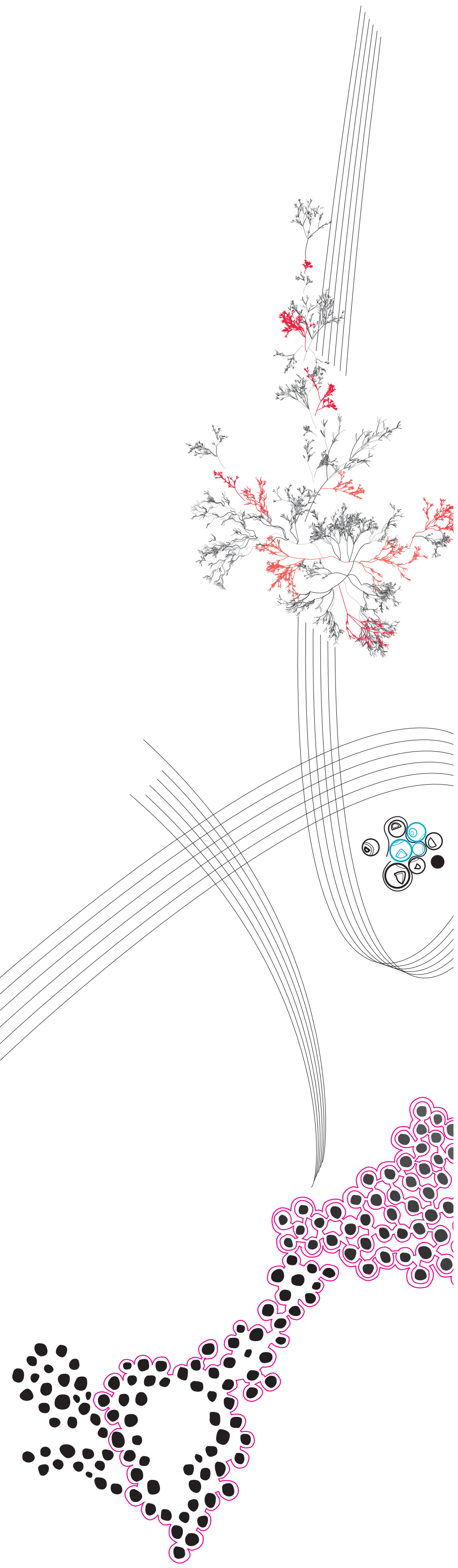
MSc Thesis Applied Mathematics



A Disjunctive Programming Approach to the Quadratic Assignment Problem Using Benders' Decomposition

David Maria van der Linden

Supervisor: M. Walter

June, 2024

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Preface

I want to thank Matthias Walter for his supervision, and his guidance during this project. Matthias, his vast knowledge of theoretical and practical mixed integer optimization, is inspiring to work with. I enjoyed diving into the mathematical details together, and then zooming out to the practical aspects of the project. I hope to work together again on another project in the future.

A Disjunctive Programming Approach to the Quadratic Assignment Problem Using Benders' Decomposition

David. M. van der Linden[*]

June, 2024

Abstract

The first version of the quadratic assignment problem (QAP) was posed by Koopmans and Beckmann in 1957. Despite the vast number of publications on the QAP, many small instances are not yet solved to optimality. Some of the most effective modern methods for solving the QAP stem from the Kaufman-Broeckx linearization. In order to advance which QAP instances can be solved, a new method for solving the QAP is proposed. This method has been derived using disjunctive programming, and Benders' decomposition. This method uses mixed integer programming and branch and cut, and this method is closely related to the Kaufman-Broeckx linearization. A prototype of the proposed method is evaluated computationally.

Keywords: Benders Decomposition, Conditional, Convex Hull, Disjunctive Programming, Facility Allocation, Mixed Integer Programming, QAP, Quadratic Assignment Problem

1 Introduction

The linear assignment problem (LAP) is one of the fundamental problems in combinatorial optimization. The LAP is the problem of assigning n facilities to n locations while minimizing a linear function of their locations. The LAP can be solved in polynomial time, using the Hungarian method [16].

The first version of the quadratic assignment problem (QAP) was posed by Koopmans and Beckmann in 1957 [15]. They presented the problem as a problem of assigning n facilities to n locations while minimizing a quadratic function of their flow and distance. Since then, the quadratic assignment problem has gained a lot of attention with over 75000 articles published[1]. The QAP has gained much attention due to its many applications and due to its resilience against solution techniques. As stated by Burkard in 2013 [5] "No exact algorithm can solve problems of size $n > 35$ in reasonable computational time nowadays". Sahni and Gonzalez have shown that finding an approximate solution within a constant factor of an optimal solution is NP-hard [24], for more information on computational complexity, we refer to [5, 23].

In this thesis, we present a new method for solving the quadratic assignment problem. This method has been derived using disjunctive programming, and Benders' decomposition. This method uses mixed integer programming and branch and cut, and this method is closely related to the Kaufman-Broeckx linearization.

[*]Email: david.van.der.linden.nl@gmail.com

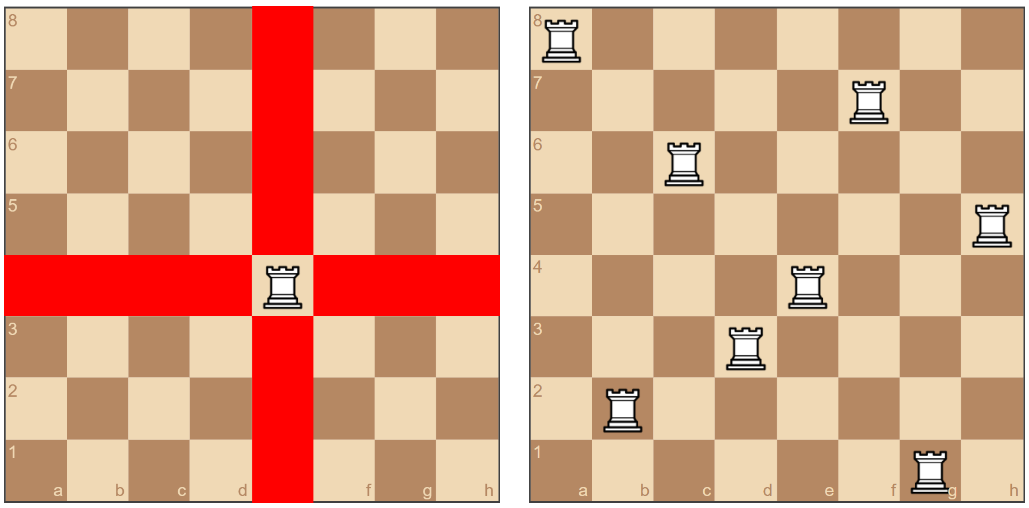
[1]See openalex.org, click the link or type Quadratic Assignment Problem in the search bar.

1.1 Problem Definition

We present a more general version of the QAP as was introduced by Lawler [17]. Let $[n]$ denote $\{1, 2, \ldots, n\}$. Given a cost matrix $q \in \mathbb{R}^{n^4}$, the Quadratic Assignment Problem is:

$$\begin{aligned}
\min \quad & \sum_{i\in[n]} \sum_{j\in[n]} \sum_{k\in[n]} \sum_{l\in[n]} q_{i,j,k,l} x_{i,j} x_{k,l} && && \text{(1a)}\\
\text{s.t.} \quad & \sum_{i\in[n]} x_{i,j} = 1 && \forall j \in [n], && \text{(1b)}\\
& \sum_{j\in[n]} x_{i,j} = 1 && \forall i \in [n], && \text{(1c)}\\
& x_{i,j} \in \{0,1\} && \forall i, j \in [n]. && \text{(1d)}
\end{aligned}$$

In this problem, there are n^2 many binary variables. These variables can be represented as an n by n grid of nodes, where each node is either on or off. With this representation, Constraints (1b) and (1c) enforce that exactly one node is on per row and per column. For $n = 8$, this problem is analogous to placing rooks on a chessboard such that no rook can attack another, as represented in Figure 1.



(A) One rook per row and column (B) A feasible assignment

Figure 1: Visualizations of row and column constraints for the QAP. Figures have been created using chessboardjs.com [22], the image of the rook was created by NikNaks [21].

Since we use these constraints a lot, let us define some shorthand for them:

Definition 1.1 ($X_n^=$ and $X_{\mathbb{R},n}^=$)**.** Let $n \in \mathbb{N}$. The set $X_n^= \subseteq \{0,1\}^{n\times n}$ is the set of vectors x satisfying

$$\begin{aligned}
& \sum_{i\in[n]} x_{i,j} = 1 && \forall j \in [n], && \text{(2a)}\\
& \sum_{j\in[n]} x_{i,j} = 1 && \forall i \in [n], && \text{(2b)}\\
& x_{i,j} \in \{0,1\} && \forall i, j \in [n]. && \text{(2c)}
\end{aligned}$$

Let $X_{\mathbb{R},n}^=$ denote the linear relaxation of $X_n^=$, meaning x satisfies $x \geq 0$, (2a), and (2b).

Now we can write the quadratic assignment problem as follows:

$$\min \sum_{i\in[n]}\sum_{j\in[n]}\sum_{k\in[n]}\sum_{l\in[n]} q_{i,j,k,l}x_{i,j}x_{k,l} \tag{3a}$$

$$\text{s.t.} \quad x \in X_n^{=}. \tag{3b}$$

In the objective, we notice parameter $q_{i,j,k,l}$ is multiplied by the product of variables $x_{i,j}$ and $x_{k,l}$. Since $x_{i,j}$ and $x_{k,l}$ are binary, the contribution to the objective of this product is either 0 or $q_{i,j,k,l}$, the latter only occurs when $x_{i,j} = x_{k,l} = 1$.

We can visualize this objective as a sum of edge weights of the complete subgraph induced in K_{n^2} by node set $\{(i,j) \mid x_{i,j} = 1\}$, see Figure 2.

Note, we may assume that the objective coefficients are non-negative, this is without loss of generality [28], since if the coefficients $q_{i,j,k,l}$ are negative, we can add a sufficiently large constant to all $q_{i,j,k,l}$. This increases the objective function by n^2 times the added constant, and does not change what the optimal permutations are.

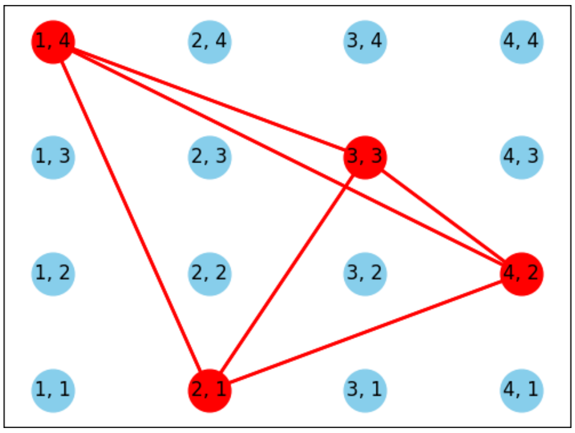


FIGURE 2: Consider the complete graph K_{n^2} with node set $\{(i,j) \mid i,j \in [n]\}$, where the edge between node (i,j) and (k,l) has edge weight $q_{i,j,k,l}$ for all $i,j,k,l \in [n]$. Then, the objective value of solution $x \in X_n^{=}$ is the sum of edge weights of the subgraph induced in K_{n^2} by $\{(i,j) \mid x_{i,j} = 1\}$.

1.2 Applications of the QAP

Many applications of the QAP are a specific type of QAP, also called the Koopmans and Beckmann Problem (KBP). The KBP is the special case of the quadratic assignment problem. Namely, the coefficients satisfy:

$$q_{i,j,k,l} = f_{i,k}d_{j,l} \qquad \forall i \in [n] \setminus \{k\}, j \in [n] \setminus \{l\}, \tag{4a}$$

$$q_{i,j,i,j} = f_{i,i}d_{j,j} + b_{i,j} \qquad \forall i,j \in [n], \tag{4b}$$

where $f, d \in \mathbb{R}^{n\times n}$ are a flow and distance matrix, and $b \in \mathbb{R}^{n^2}$ is a linear cost term. In many applications $f_{i,i}$, $d_{j,j}$ and $b_{i,j}$ are zero for all $(i,j) \in [n] \times [n]$.

One application is the plumbing problem with n locations and n machines. One has to find an appropriate location for each machine as to minimize the total cost of pipes required. Here, machine i has a fixed flow of liquid that goes to machine k. This flow corresponds to a certain diameter of pipe, and has a cost $f_{i,k}$ of that pipe per meter. Each location pair (j,l) has a distance $d_{j,l}$ in meters between them. This yields the cost $q_{i,j,k,l}$

of the pipe going from machine i at location j to machine k at location l. The cost $q_{i,j,i,j}$ is the cost of building machine i at location j.

The plumbing problem was designed for the sake of this article, but is inspired by Steinberg's backboard wiring problem [25, 6]. The backboard wiring problem has the objective to minimize the total wire length. Here $d_{j,l}$ represents the distance from wire j to wire l, and $f_{i,k} = 1$ if there is a wire connecting device i to device k, otherwise $f_{i,k} = 0$.

Real world examples of QAP applications are: typewriter keyboard design [4], placement of electronic components [20], campus building arrangement [8], economic problems (capturing the largest market share) [11], and many more. For further reading on applications of the QAP see [18].

1.3 The Kaufman-Broeckx Family of Linearizations

Quadratic problems with integrality constraints are typically quite hard to solve. However, solving linear problems with integrality constraints can be done for reasonable problem sizes using modern mixed integer linear solvers. For this reason, we will consider linearizations of the QAP.

In this section, we look at two linearizations of the QAP: the Kaufman-Broeckx Linearization and the Xia-Yuan Linearization. The Xia-Yuan Linearization uses the Gilmore-Lawler bound. This bound can also be used to linearize the QAP, however we only introduce the Gilmore-Lawler bound in order to understand the Xia-Yuan Linearization.

1.3.1 Kaufman-Broeckx Linearization

Kaufman and Broeckx [14] introduce a variable $\tilde{w}_{i,j} = x_{i,j} \sum_{k\in[n]} \sum_{l\in[n]} q_{i,j,k,l} x_{k,l}$, and use a big-M constraint to linearize $\tilde{w}$. The Kaufman-Broeckx linearization is

$$\min \sum_{i\in[n]} \sum_{j\in[n]} \tilde{w}_{i,j} \tag{5a}$$

$$\text{s.t.} \quad \tilde{w}_{i,j} \geq \sum_{k\in[n]} \sum_{l\in[n]} q_{i,j,k,l} x_{k,l} - \tilde{M}_{i,j}(1 - x_{i,j}) \quad \forall i,j \in [n], \tag{5b}$$

$$\tilde{w}_{i,j} \geq 0 \quad \forall i,j \in [n], \tag{5c}$$

$$x \in X_n^=, \tag{5d}$$

where $\tilde{M}_{i,j} = \sum_{k\in[n]} \sum_{l\in[n]} q_{i,j,k,l}$. Notice how, since $\tilde{M}_{i,j}$ is sufficiently large, constraint (5b) are essentially deactivated when $x_{i,j} = 0$.

1.3.2 Gilmore-Lawler Bound

The Gilmore-Lawler bound is a lower bound on the QAP and was found independently by Gilmore [13] and Lawler [17]; see [28]. The Gilmore-Lawler bound is:

$$\min \sum_{i\in[n]} \sum_{j\in[n]} (L_{i,j} + q_{i,j,i,j}) x_{i,j} \tag{6a}$$

$$\text{s.t.} \quad x \in X_n^=, \tag{6b}$$

where $L_{i,j}$ is the optimal value of the following linear assignment problem:

$$\min \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l} x_{k,l} \tag{7a}$$

$$\text{s.t.} \quad x \in X_n^=. \tag{7b}$$

Finding $L_{i,j}$ requires solving a linear assignment problem per (i,j) pair in $[n] \times [n]$. Solving a linear assignment problem can be done in polynomial time, using the Hungarian method [16].

1.3.3 Xia-Yuan Linearization

Xia and Yuan have strengthened the Kaufman-Broeckx Linearization [27, 26]; see [28], by adjusting the big-M term, the 0 term, and excluding $q_{i,j,i,j}x_{i,j}$ from $\bar{w}_{i,j}$, inspired by the Gilmore-Lawler bound, creating another formulation for the QAP. The Xia-Yuan linearization is

$$\min \sum_{i\in[n]} \sum_{j\in[n]} (\bar{w}_{i,j} + q_{i,j,i,j}x_{i,j}) \tag{8a}$$

$$\text{s.t.} \quad \bar{w}_{i,j} \geq \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l}x_{k,l} - \bar{M}_{i,j}(1 - x_{i,j}) \quad \forall i,j \in [n], \tag{8b}$$

$$\bar{w}_{i,j} \geq L_{i,j}x_{i,j} \quad \forall i,j \in [n], \tag{8c}$$

$$x \in X_n^=, \tag{8d}$$

where $\bar{M}_{i,j} = \max(\sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l}x_{k,l} \mid x \in X_n^=)$ for $i,j \in [n]$, and $L_{i,j}$ the optimal value of (7).

1.4 Overview of Proposed Method

We consider the QAP where $q \in \mathbb{R}^{n^4}$ is the cost matrix. We introduce variables $w_{i,j}$ that are the sum of costs of all incident edges $\{(i,j),(k,l)\}$ for all $k,l \in [n]$.

$$w_{i,j} = \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l}x_{k,l} \quad \text{if } x_{i,j} = 1, \tag{9a}$$

$$w_{i,j} = 0 \quad \text{if } x_{i,j} = 0. \tag{9b}$$

We refer to $w_{i,j}$ as the *node contribution* of node $x_{i,j}$. Now that we have introduced these variables, the objective of the QAP can be rewritten to minimize $\sum_i \sum_j (w_{i,j} + q_{i,j,i,j}x_{i,j})$. When looking at a specific $w_{i,j}$, we can linearize the system by considering two cases $x_{i,j} = 0$ or $x_{i,j} = 1$. These two cases can be described using linear inequalities and will define polytopes $P_{i,j}^{(0)}$ and $P_{i,j}^{(1)}$ respectively. If we have inequalities defining the convex hull of $P_{i,j}^{(0)} \cup P_{i,j}^{(1)}$ then we might be able to speed up the solving process of the main problem. In the Kaufman-Broeckx linearization, there is a non-negativity constraint on $w_{i,j}$ and a big-M constraint. While Fischetti et al. [10] have provided insight into the utility of the convex hull $P_{i,j}$ by means of strengthening big-M constraints, they have not proposed a method of generating the convex hull nor applied a method that utilizes multiple constraints per $w_{i,j}$ variable. We intend to generate more inequalities in the hope that this will provide a significant speedup of the solving process. We will use Disjunctive programming to find a system of inequalities that defines polytope $P_{i,j}$, where the projection of $P_{i,j}$ onto x is the convex hull of $P_{i,j}^{(0)}$ and $P_{i,j}^{(1)}$. Since this projection depends on objective coefficients $q \in \mathbb{R}^{n^4}$, carrying the projection out by hand using Fourier–Motzkin elimination seems impractical. Benders' decomposition can produce inequalities of a projection algorithmically. We will use Benders' decomposition in order to generate inequalities that define the projection of $P_{i,j}$ onto x.

2 Method and Derivations

Consider the QAP where $q \in \mathbb{R}^{n^4}$ is the cost matrix. Consider variables $w_{i,j}$ that are the sum of costs of all incident edges $((i,j),(k,l))$ for all $k,l \in [n]$.

$$w_{i,j} = \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l} x_{k,l} \qquad \text{if } x_{i,j} = 1, \tag{10a}$$

$$w_{i,j} = 0 \qquad \text{if } x_{i,j} = 0. \tag{10b}$$

Then the objective of the QAP can be rewritten to minimize $\sum_i \sum_j (w_{i,j} + q_{i,j,i,j} x_{i,j})$. The Kaufman-Broeckx linearization works by using big-M constraints in order to impose the condition $w_{i,j} = \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l} x_{k,l}$ if $x_{i,j} = 1$. We will use this node contribution variable $w_{i,j}$ with a different approach. Namely, by applying disjunctive programming, which at its core is using the following theorem:

Theorem 1 (Balas [2]; see [7]). *Let $A \in \mathbb{R}^{n\times m_1}$, $B \in \mathbb{R}^{n\times m_2}$, $n, m_1, m_2 \in \mathbb{Z}_{>0}$, and $b, d \in \mathbb{R}^n$. Given two polytopes*

$$P^{(0)} = \{x^{(0)} : Ax^{(0)} \le b\} \subseteq \mathbb{R}^n, \tag{11a}$$

$$P^{(1)} = \{x^{(1)} : Bx^{(1)} \le d\} \subseteq \mathbb{R}^n. \tag{11b}$$

Then conv$(P^{(0)} \cup P^{(1)})$ is the projection of the polytope

$$\{(x^{(0)}, x^{(1)}, \alpha, \beta, x) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}^n \tag{12a}$$

$$\text{such that } Ax^{(0)} \le \alpha b \tag{12b}$$

$$Bx^{(1)} \le \beta d \tag{12c}$$

$$\alpha + \beta = 1 \tag{12d}$$

$$\alpha, \beta \ge 0 \tag{12e}$$

$$x = x^{(0)} + x^{(1)}\} \tag{12f}$$

onto x, where conv$(\cdot)$ denotes the convex hull.

Now, consider (x, w_{ij}), where $x \in X^{=}_{\mathbb{R},n}$. Specifically, let us look at:

$$P^{(0)}_{i,j} = \{(x, 0) \mid x_{ij} = 0 \text{ and } x \in X^{=}_{\mathbb{R},n}\}, \tag{13a}$$

$$P^{(1)}_{i,j} = \{(x, w_{ij}) \mid x_{ij} = 1, x \in X^{=}_{\mathbb{R},n}, \text{ and (10a) holds}\}. \tag{13b}$$

Now we will use disjunctive programming to describe $P_{i,j} = \text{conv}(P^{(0)}_{i,j} \cup P^{(1)}_{i,j})$.

Theorem 2. *The polytope $P_{i,j}$ is described by projection of* (14) *onto $(x, w_{i,j})$.*

$$x_{i,j} \le 1, \tag{14a}$$

$$x_{i,j} \ge 0, \tag{14b}$$

$$x_{k,l} - x^{(1)}_{k,l} \ge 0 \qquad \forall k \in [n] \setminus \{i\}, l \in [n] \setminus \{j\}, \tag{14c}$$

$$\sum_{k\in[n]\setminus\{i\}} x^{(1)}_{k,l} = x_{i,j} \qquad \forall l \in [n] \setminus \{j\}, \tag{14d}$$

$$\sum_{l\in[n]\setminus\{j\}} x^{(1)}_{k,l} = x_{i,j} \qquad \forall k \in [n] \setminus \{i\}, \tag{14e}$$

$$x^{(1)}_{k,l} \ge 0 \qquad \forall k \in [n] \setminus \{i\}, l \in [n] \setminus \{j\}, \tag{14f}$$

$$w_{i,j} = \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l} x^{(1)}_{k,l}. \tag{14g}$$

Proof. The polytope $P_{i,j}^{(0)} \subset \mathbb{R} \times \mathbb{R}^{n\times n}$ is defined by the following constraints:

$$\begin{aligned}
x_{i,j}^{(0)} &= 0, && && \text{(15a)}\\
w_{i,j}^{(0)} &= 0, && && \text{(15b)}\\
\sum_{k\in[n]} x_{k,l}^{(0)} &= 1 && \forall l \in [n], && \text{(15c)}\\
\sum_{l\in[n]} x_{k,l}^{(0)} &= 1 && \forall k \in [n], && \text{(15d)}\\
x_{k,l}^{(0)} &\geq 0 && \forall k, l \in [n]. && \text{(15e)}
\end{aligned}$$

Polytope $P_{i,j}^{(1)} \subset \mathbb{R} \times \mathbb{R}^{n\times n}$ is defined by the following constraints:

$$\begin{aligned}
x_{i,j}^{(1)} &= 1, && && \text{(16a)}\\
x_{k,j}^{(1)} &= 0 && \forall k \in [n] \setminus \{i\}, && \text{(16b)}\\
x_{i,l}^{(1)} &= 0 && \forall l \in [n] \setminus \{j\}, && \text{(16c)}\\
w_{i,j}^{(1)} - \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l} x_{k,l}^{(1)} &= 0, && && \text{(16d)}\\
\sum_{k\in[n]\setminus\{i\}} x_{k,l}^{(1)} &= 1 && \forall l \in [n] \setminus \{j\}, && \text{(16e)}\\
\sum_{l\in[n]\setminus\{j\}} x_{k,l}^{(1)} &= 1 && \forall k \in [n] \setminus \{i\}, && \text{(16f)}\\
x_{k,l}^{(1)} &\geq 0 && \forall k, l \in [n]. && \text{(16g)}
\end{aligned}$$

Now we want to apply disjunctive programming, Theorem 1, to these two polytopes in order get their convex hull. Since $\beta = 1 - \alpha$ we can bound α from below and above and leave out β. Applying Theorem 1 yields the following system for $P_{i,j}$:

$$0 \le \alpha \le 1, \tag{17a}$$
$$x^{(0)}_{i,j} = 0, \tag{17b}$$
$$w^{(0)}_{i,j} = 0, \tag{17c}$$
$$\sum_{k\in[n]} x^{(0)}_{k,l} = \alpha \qquad \forall l \in [n], \tag{17d}$$
$$\sum_{l\in[n]} x^{(0)}_{k,l} = \alpha \qquad \forall k \in [n], \tag{17e}$$
$$x^{(0)}_{k,l} \ge 0 \qquad \forall k,l \in [n], \tag{17f}$$
$$x^{(1)}_{i,j} = 1-\alpha, \tag{17g}$$
$$x^{(1)}_{k,j} = 0 \qquad \forall k \in [n]\setminus\{i\}, \tag{17h}$$
$$x^{(1)}_{i,l} = 0 \qquad \forall l \in [n]\setminus\{j\}, \tag{17i}$$
$$w^{(1)}_{i,j} - \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l} x^{(1)}_{k,l} = 0, \tag{17j}$$
$$\sum_{k\in[n]\setminus\{i\}} x^{(1)}_{k,l} = 1-\alpha \qquad \forall l \in [n]\setminus\{j\}, \tag{17k}$$
$$\sum_{l\in[n]\setminus\{j\}} x^{(1)}_{k,l} = 1-\alpha \qquad \forall k \in [n]\setminus\{i\}, \tag{17l}$$
$$x^{(1)}_{k,l} \ge 0 \qquad \forall k,l \in [n], \tag{17m}$$
$$x_{k,l} = x^{(0)}_{k,l} + x^{(1)}_{k,l} \qquad \forall k,l \in [n], \tag{17n}$$
$$w_{i,j} = w^{(0)}_{i,j} + w^{(1)}_{i,j}. \tag{17o}$$

Combining (17o), (17c) and (17j) yields $w_{i,j} = \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l} x^{(1)}_{k,l}$. Combining (17n) for $k = i, l = j$ with (17b) and (17g) yields $x_{i,j} = 1-\alpha$. We can substitute

$x_{i,j}^{(0)}$ by $x_{i,j} - x_{i,j}^{(1)}$. Rewriting yields us the following system for $P_{i,j}$:

$$
\begin{aligned}
x_{i,j} &\leq 1, && && \text{(18a)}\\
x_{i,j} &\geq 0, && && \text{(18b)}\\
\sum_l (x_{k,l} - x_{k,l}^{(1)}) &= 1 - x_{i,j} && \forall k, && \text{(18c)}\\
\sum_k (x_{k,l} - x_{k,l}^{(1)}) &= 1 - x_{i,j} && \forall l, && \text{(18d)}\\
x_{k,l} - x_{k,l}^{(1)} &\geq 0 && \forall k, l, && \text{(18e)}\\
x_{i,j}^{(1)} &= x_{i,j}, && && \text{(18f)}\\
x_{k,j}^{(1)} &= 0 && \forall k \in [n] \setminus \{i\}, && \text{(18g)}\\
x_{i,l}^{(1)} &= 0 && \forall l \in [n] \setminus \{j\}, && \text{(18h)}\\
\sum_{k \in [n] \setminus \{i\}} x_{k,l}^{(1)} &= x_{i,j} && \forall l \in [n] \setminus \{j\}, && \text{(18i)}\\
\sum_{l \in [n] \setminus \{j\}} x_{k,l}^{(1)} &= x_{i,j} && \forall k \in [n] \setminus \{i\}, && \text{(18j)}\\
x_{k,l}^{(1)} &\geq 0 && \forall k, l, && \text{(18k)}\\
w_{i,j} &= \sum_{k \in [n] \setminus \{i\}} \sum_{l \in [n] \setminus \{j\}} q_{i,j,k,l} x_{k,l}^{(1)}. && && \text{(18l)}
\end{aligned}
$$

Since the sum of a row or column is 1, (18c) and (18d) simplify to (18i) and (18j) respectively. Moreover, for $(k, l) = (i, j)$, (18e) and (18k) is implied by (18f), (18g) and (18h). This means the system be simplified to:

$$
\begin{aligned}
x_{i,j} &\leq 1, && && \text{(19a)}\\
x_{i,j} &\geq 0, && && \text{(19b)}\\
x_{k,l} - x_{k,l}^{(1)} &\geq 0 && \forall k \in [n] \setminus \{i\}, l \in [n] \setminus \{j\}, && \text{(19c)}\\
x_{i,j}^{(1)} &= x_{i,j}, && && \text{(19d)}\\
x_{k,j}^{(1)} &= 0 && \forall k \in [n] \setminus \{i\}, && \text{(19e)}\\
x_{i,l}^{(1)} &= 0 && \forall l \in [n] \setminus \{j\}, && \text{(19f)}\\
\sum_{k \in [n] \setminus \{i\}} x_{k,l}^{(1)} &= x_{i,j} && \forall l \in [n] \setminus \{j\}, && \text{(19g)}\\
\sum_{l \in [n] \setminus \{j\}} x_{k,l}^{(1)} &= x_{i,j} && \forall k \in [n] \setminus \{i\}, && \text{(19h)}\\
x_{k,l}^{(1)} &\geq 0 && \forall k \in [n] \setminus \{i\}, l \in [n] \setminus \{j\}, && \text{(19i)}\\
w_{i,j} &= \sum_{k \in [n] \setminus \{i\}} \sum_{l \in [n] \setminus \{j\}} q_{i,j,k,l} x_{k,l}^{(1)}. && && \text{(19j)}
\end{aligned}
$$

Since variables $x_{k,l}^{(1)}$ for which $k = i$ or $l = j$ only occur in constraints where a single variable is set to $x_{i,j}$ or 0, and do not occur in the objective, equations (19d), (19e) and (19f) can be dropped. This yields the system for $P_{i,j}$.

$\square$

System (14) together with the appropriate objective can be reduced to a Minimum Constrained Flow Problem. More on this in Section 3.

2.1 Applying Benders' Decomposition

In this subsection, we apply Benders' decomposition [3] in order to carry out the projection of $P_{i,j}$ onto $(x, w_{i,j})$ as described by Theorem 2.

Common terminology Let us start by introducing some common terminology. When adding inequality to a system of equations, the region of feasible solutions can become smaller. Such an inequality is called a *cutting plane* or *cut* for short. A linear program (LP) with integrality constraints is referred to as a *mixed integer linear program* (MILP). The *linear relaxation* or *LP relaxation* of a MILP is the MILP without its integrality constraints. A *separation algorithm* is an algorithm that separates feasible from infeasible solutions, by means of generating a cutting plane when an infeasible solution is presented.

Branch and bound *Branch and bound* is a method for solving MILPs. Branch and bound works by solving the linear relaxation, this yields a solution. This solution together with its problem is called a branch and bound *node*. If a branch and bound node does not satisfy all integrality constraints, we can *branch* on this node by using rounding to create sub-problems.

We now present an example of rounding to create sub-problems. Suppose, we have a branch and bound node with solution $\hat{x}$ and entry $\hat{x}_i = 0.5$ while the MILP has constraint, $\hat{x}_i \in \mathbb{Z}$ then we know that any feasible MILP solution satisfies the constraint: $x_i \leq \lfloor 0.5 \rfloor = 0$ or the constraint: $x_i \geq \lceil 0.5 \rceil = 1$. These constraints each correspond to a new sub-problem in the branch and bound tree.

When a branch and bound node solution satisfies all the integrality constraints, it is a MILP solution. The objective value of MILP solutions and LP relaxation solutions at nodes, can be used to navigate the branch and bound tree efficiently and find the optimum solution of the MILP. For more information on branch and bound, we refer to [7]. When branch and bound is used in combination with cutting planes, the procedure is called *branch and cut*. Cutting planes are generated in order to speed up the branch and bound process.

Types of problems While solving a specific QAP instance, we deal with three types of problems:

1. The QAP, with a quadratic objective and mixed integer linear constraints,

2.1. The *main MILP*, solving a MILP requires solving its LP relaxation,

2.2. the main MILP relaxation, often referred to as the *main LP*,

3. and many *sub-problems*, also referred to as *lifting LPs*.

Callback In this thesis, calling *the separation algorithm* means applying Theorem 4 to prove a solution satisfies the conditions we want to impose on w or provide one or several cuts that separate this solution from the feasible solutions. The separation algorithm is called within a callback. A *callback* is a customizable function within a MILP solver. We can customize the callback by:

1. setting the callback to call the separation algorithm at each node of the MILP branch and cut tree, or by
2. setting the callback to call the separation algorithm at each MILP solution. These MILP solutions can be attained from branch and cut nodes, or from heuristic attempts at an optimal solution.

High Level Process see Figure 3. First, we initialize the main MILP. Secondly, we solve the main MILP while using a set callback. If the separation algorithm is called, the current main MILP or main LP solution $(\hat{x}, \hat{w})$ is available. This solution $(\hat{x}, \hat{w})$ is then used to create and solve one sub-problem per (i, j) pair. From the solutions of the sub-problems we conclude that we found a feasible and optimal solution to the QAP, or use the solutions of the sub-problems to adjust the main MILP by adding cuts. When branch and cut solves the main MILP, assisted by the cutting planes from the callback, we have an optimal solution to the QAP.

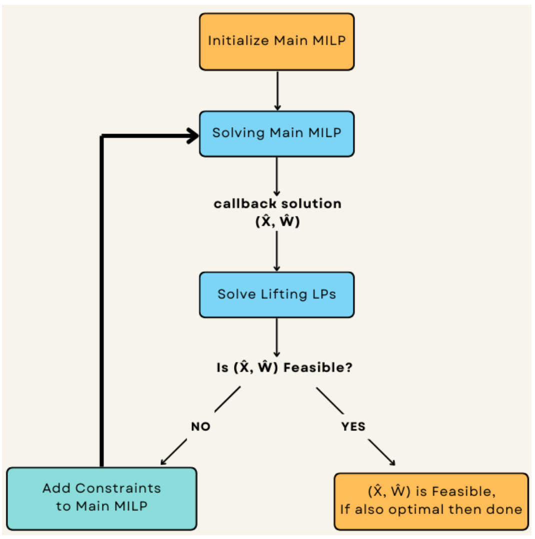


FIGURE 3: High Level Process

Main MILP

$$\min \sum_{i\in[n]} \sum_{j\in[n]} (w_{i,j} + q_{i,j,i,j} x_{i,j}) \tag{20a}$$

$$\text{s.t.} \quad A \begin{pmatrix} x \\ w \end{pmatrix} \geq b, \tag{20b}$$

$$x \in X_n^{=}, \tag{20c}$$

$$w \in \mathbb{R}^{n^2}, \tag{20d}$$

where matrix A and vector b ensure (20b) captures all constraints on w. We initialize A and b as to consist of only the inequality $w \geq 0$, and update A and b when the separation algorithm provides a cutting plane. As explained in Section 1.1, we may assume that q is non-negative, justifying the constraint $w \geq 0$. Additional initialization could be adding all Kaufman-Broeckx constraints or all Xia-Yuan constraints.

2.1.1 Lifting LP Derivations

At the core of the Benders' decomposition approach, we are given some $\hat{x} \in \mathbb{R}^{n\times n}$ and $\hat{w} \in \mathbb{R}^{n\times n}$, obtained by solving the main LP (20). We want to check per combination of (i, j) if $(\hat{x}, \hat{w}_{i,j})$ are in the convex hull of $P_{i,j}^{(1)} \cup P_{i,j}^{(0)}$, otherwise, generate a cutting plane in order to generate a cutting plane, we have to solve a sub-problem.

Solving the main LP (20) yields some solution $x = \hat{x}, w = \hat{w}$. Applying the theorem by Balas as in the previous section yields (14), as described in Theorem 2. Due to the constraints on the main problem, constraints (14a) and (14b) are satisfied. Getting the sub-problem for Benders' decomposition of (i, j) in this iteration works by fixing the $x_{k,l}$ variables to $\hat{x}_{k,l}$ for all $k \in [n] \setminus \{i\}, l \in [n] \setminus \{j\}$, and minimizing for

$w_{i,j} = \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l} x^{(1)}_{k,l}$. This yields the following sub-problem:

$$\min \; w_{i,j} = \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l} x^{(1)}_{k,l} \tag{21a}$$

$$\text{s.t.} \quad x^{(1)}_{k,l} \le \hat{x}_{k,l} \quad \forall k \in [n]\setminus\{i\}, l \in [n]\setminus\{j\}, \tag{21b}$$

$$\sum_{k\in[n]\setminus\{i\}} x^{(1)}_{k,l} = \hat{x}_{i,j} \quad \forall l \in [n]\setminus\{j\}, \tag{21c}$$

$$\sum_{l\in[n]\setminus\{j\}} x^{(1)}_{k,l} = \hat{x}_{i,j} \quad \forall k \in [n]\setminus\{i\}, \tag{21d}$$

$$x^{(1)}_{k,l} \ge 0 \quad \forall k \in [n]\setminus\{i\}, l \in [n]\setminus\{j\}. \tag{21e}$$

We also refer to this sub-problem as the lifting LP. The lifting LP has the following dual LP:

$$\max \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} \hat{x}_{k,l}\lambda_{k,l} + x_{i,j} \sum_{k\in[n]\setminus\{i\}} \phi_k + x_{i,j} \sum_{l\in[n]\setminus\{j\}} \theta_l \tag{22a}$$

$$\text{s.t.} \quad \lambda_{k,l} \le 0 \quad \forall k \in [n]\setminus\{i\}, l \in [n]\setminus\{j\}, \tag{22b}$$

$$\lambda_{k,l} + \phi_k + \theta_l \le q_{i,j,k,l} \quad \forall k \in [n]\setminus\{i\}, l \in [n]\setminus\{j\}. \tag{22c}$$

For literature on duality, see [19].

Suppose the resulting optimal value of the lifting LP (21) is $\bar{w}_{i,j} \in \mathbb{R}$. If $\hat{w}_{i,j} < \bar{w}_{i,j} < \infty$, then the main LP is still missing constraints and yields an unrealistically low objective value. We use Farkas' Lemma to find a constraint that sharpens the conditions on the objective variables. This type of constraint is called an *optimality cut.*

Theorem 3 (Farkas' Lemma [9]; see [12])**.** *Let $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. Then exactly one of the following two assertions is true:*

1. *There exists an $x \in \mathbb{R}^n$ such that $Ax = b$ and $x \ge 0$.*

2. *There exists a $y \in \mathbb{R}^m$ such that $A^\intercal y \ge 0$ and $b^\intercal y < 0$.*

So either there exists a feasible solution to the primal problem or an unbounded improving direction in the dual that is also called a *Farkas' ray.* Applying Farkas' Lemma, Theorem 3, allows us to prove Theorem 4.

Theorem 4. *Let the linear relaxation of the main MILP have solution $(\hat{x}, \hat{w})$, let the lifting LP* (21) *have minimum value $\bar{w}_{i,j}$, and dual multipliers $\bar{\lambda}$, $\bar{\theta}$, and $\bar{\phi}$ for constraints* (21b), (21c)*, and* (21d) *respectively. If $\hat{w}_{i,j} < \bar{w}_{i,j}$, then*

$$w_{i,j} \ge \Big(\sum_{l\in[n]\setminus\{j\}} \bar{\theta}_l + \sum_{k\in[n]\setminus\{i\}} \bar{\phi}_k\Big) x_{i,j} + \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} \bar{\lambda}_{k,l} x_{k,l}, \tag{23}$$

is a valid inequality for the IP formulation of the main MILP, and cuts off the current solution of the linear relaxation of the main MILP.

Otherwise $\hat{w}_{i,j} \ge \bar{w}_{i,j}$, and then $(\hat{x}, \hat{w}_{i,j})$ belongs to $P_{i,j}$.

Proof. Consider the situation as described by the theorem and let $(\hat{x}, \hat{w})$, $\bar{\lambda}$, $\bar{\theta}$, and $\bar{\phi}$ be defined as in the theorem. We start by proving the latter statement. Suppose $\hat{w}_{i,j} \ge \bar{w}_{i,j}$. Then there exists a vector $\hat{x}^{(1)} \in \mathbb{R}^{n\times n}$ such that the vector $(\hat{x}, \hat{w}, \hat{x}^{(1)})$ satisfies (14).

By the Balas' theorem, Theorem 1, the projection of $(\hat{x}, \hat{w}, \hat{x}^{(1)})$ onto $(x, w_{i,j})$, which is $(\hat{x}, \hat{w}_{i,j})$, belongs to $P_{i,j}$.

Now, suppose $\hat{w}_{i,j} < \bar{w}_{i,j}$. We create a system from the constraints of (21) and add one constraint to it, resulting in the following system:

$$\sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l} x^{(1)}_{k,l} \leq \hat{w}_{i,j} \tag{24a}$$

$$x^{(1)}_{k,l} \leq \hat{x}_{k,l} \qquad \forall k \in [n] \setminus \{i\}, l \in [n] \setminus \{j\}, \tag{24b}$$

$$\sum_{k\in[n]\setminus\{i\}} x^{(1)}_{k,l} = \hat{x}_{i,j} \qquad \forall l \in [n] \setminus \{j\}, \tag{24c}$$

$$\sum_{l\in[n]\setminus\{j\}} x^{(1)}_{k,l} = \hat{x}_{i,j} \qquad \forall k \in [n] \setminus \{i\}, \tag{24d}$$

$$x^{(1)}_{k,l} \geq 0 \qquad \forall k \in [n] \setminus \{i\}, l \in [n] \setminus \{j\}. \tag{24e}$$

Rewriting (24) to standard form and negating some equations yields:

$$\sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l} x^{(1)}_{k,l} + T = \hat{w}_{i,j} \tag{25a}$$

$$-x^{(1)}_{k,l} - S_{i,j} = -\hat{x}_{k,l} \qquad \forall k \in [n] \setminus \{i\}, l \in [n] \setminus \{j\}, \tag{25b}$$

$$-\sum_{k\in[n]\setminus\{i\}} x^{(1)}_{k,l} = -\hat{x}_{i,j} \qquad \forall l \in [n] \setminus \{j\}, \tag{25c}$$

$$-\sum_{l\in[n]\setminus\{j\}} x^{(1)}_{k,l} = -\hat{x}_{i,j} \qquad \forall k \in [n] \setminus \{i\}, \tag{25d}$$

$$x^{(1)} \geq 0, \tag{25e}$$

$$T \geq 0, \tag{25f}$$

$$S \geq 0, \tag{25g}$$

where $T \in \mathbb{R}$ and $S \in \mathbb{R}^{(n-1)^2}$ are slack variables. We now apply Farkas' Lemma to system (25). Since $\hat{w}_{i,j} < \bar{w}_{i,j}$, and $\bar{w}$ was already the minimum of (21), we know that constraint (24a) can not be satisfied while satisfying the other constraints of (24). Proving system (24) is infeasible. Therefore, system (25) is not feasible either. This implies system (25) is not feasible. Consequently, by Farkas' Lemma there exists a Farkas' ray. Looking at our system, we can see this means there exist $\lambda \in \mathbb{R}^{n\times n}, \theta \in \mathbb{R}^n, \phi \in \mathbb{R}^n$, and $\alpha \in \mathbb{R}$ such that:

$$\hat{w}_{i,j}\alpha - \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} \hat{x}_{k,l}\lambda_{k,l} - \hat{x}_{i,j} \sum_{l\in[n]\setminus\{j\}} \theta_l - \hat{x}_{i,j} \sum_{k\in[n]\setminus\{i\}} \phi_k < 0, \tag{26a}$$

$$q_{i,j,k,l}\alpha - \lambda_{k,l} - \theta_l - \phi_k \geq 0 \qquad \forall k, l \in [n], \tag{26b}$$

$$\lambda \leq 0, \tag{26c}$$

$$\alpha \geq 0. \tag{26d}$$

Since all constraints of system (26) can still be scaled, we normalize the system by setting $\alpha = 1$. This yields:

$$\hat{w}_{i,j} - \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} \hat{x}_{k,l}\lambda_{k,l} - \hat{x}_{i,j} \sum_{l\in[n]\setminus\{j\}} \theta_l - \hat{x}_{i,j} \sum_{k\in[n]\setminus\{i\}} \phi_k < 0, \tag{27a}$$

$$q_{i,j,k,l} - \lambda_{k,l} - \theta_l - \phi_k \geq 0 \qquad \forall k, l \in [n], \tag{27b}$$

$$\lambda \leq 0. \tag{27c}$$

Let us now consider the following inequality:

$$w_{i,j} - (\sum_{l\in[n]\setminus\{j\}} \bar{\theta}_l + \sum_{k\in[n]\setminus\{i\}} \bar{\phi}_k)x_{i,j} - \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} \bar{\lambda}_{k,l}x_{k,l} \geq 0. \tag{28}$$

If we minimize the left hand side of (27a) subject to (27b) and (27c), then we get the coefficients for the cut with maximal cut violation. We claim without proof that solving the dual of the lifting LP (22) yields the same coefficients ($\bar{\lambda}$, $\bar{\theta}$, and $\bar{\phi}$) and therefore yields the maximum violated cut.

We now prove that (28) is valid for $P_{i,j}$ by using Farkas' Lemma. Suppose we have a solution $(\tilde{x}, \tilde{w}_{i,j}) \in P_{i,j}$. This implies system (25) has a feasible solution. By Farkas' Lemma there does not exist a Farkas' ray, i.e., System (27) is feasible. Since $\bar{\lambda}$, $\bar{\theta}$, and $\bar{\phi}$ are obtained from the dual they satisfy the latter constraints of (27) implying the in-feasibility must follow from (27a) further implying (28) is a valid inequality for $P_{i,j}$.

As for proving that this cuts off the current solution: Since $(\tilde{x}, \tilde{w}_{i,j})$ satisfied (27a) adding (28) cuts off $(\hat{x}, \hat{w})$. □

Looking Back at the High Level Process The aim of this paragraph is to support Figure 3 and summarize the findings of in this section. If the separation algorithm is called, the current main MILP or main LP solution $(\hat{x}, \hat{w})$ is available. This solution $(\hat{x}, \hat{w})$ is then used to create and solve one sub-problem per (i, j) pair. Let $\bar{w}_{i,j}$ be the solution of sub-problem (i, j).

If $\hat{w}_{i,j} \geq \bar{w}_{i,j}$ for all $(i, j) \in [n] \times [n]$, then $(\hat{x}, \hat{w}_{i,j}) \in P_{i,j}$ for all $(i, j) \in [n] \times [n]$, by Theorem 4. Meaning, our current solution $(\hat{x}, \hat{w})$ has the property that all $w_{i,j}$ are some convex combination of other values $w'_{i,j}, \ldots$ satisfying integrality of x and the node contribution relation of w and x as defined by (9).

If $\hat{w}_{i,j} < \bar{w}_{i,j}$ for some $(i, j) \in [n] \times [n]$, then constraint (23) can be added to the main MILP cutting of the current solution $(\hat{x}, \hat{w})$, see Theorem 4. If the solution $(\hat{x}, \hat{w})$ was from a callback at a branch and cut node, then this cut might speed up the solving process. If the solution $(\hat{x}, \hat{w})$ was from a callback at a MILP solution, this cut is required for ensuring the MILP solution objective of $(\hat{x}, \hat{w})$ is the same as the QAP objective of $\hat{x}$.

3 Speeding Up the Lifting LP

We claim without proof that the lifting LP (21) can be reduced to a minimum cost flow problem. There are several combinatorial algorithms that solve the minimum cost flow problem. Combinatorial algorithms are typically much faster at solving problems than solving the same problem using an LP formulation. Since we aim to extract dual information from the lifting LP, we recommend solving the LP using a primal-dual algorithm as described in [1].

4 Experiments

4.1 Experimental Setup

Experiments have been implemented in Python using Gurobipy. Gurobipy is the Python interface for the optimization software called Gurobi, which uses branch and cut to optimize MILPs.

KBL The implementation of the Kaufman-Broeckx linearization, further referred to as *KBL*, is done by first specifying the variables, constraints, and objective as described in Section 1.3.1; Secondly, adjusting some Gurobi optimization parameters, more on these parameters later; And finally, solving the problem using Gurobi.

DP The implementation of the method proposed in this article, further referred to as *DP*, is less trivial. The main MILP is implemented as described in by System (20), where A and b are initialized to contain the following constraints:

$$w_{i,j} \geq \sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l}x_{k,l} - M_{i,j}(1 - x_{i,j}) \qquad \forall i, j \in [n], \tag{29a}$$

where $M_{i,j} = \tilde{M}_{i,j} = \sum_{k\in[n]} \sum_{l\in[n]} q_{i,j,k,l}$, these are Kaufman-Broeckx constraints with a minor adjustment. The adjustment is to account for the altered objective function, w is without the $q_{i,j,i,j}x_{i,j}$ term while $\tilde{w}_{i,j}$ is with the $q_{i,j,i,j}x_{i,j}$ term, see Section 1.3.1 and Equation (10a). Note $M_{i,j}$ could have been, but is not set to: $\sum_{k\in[n]\setminus\{i\}} \sum_{l\in[n]\setminus\{j\}} q_{i,j,k,l}$ which would strengthen the constraint whenever $q_{i,j,i,j} \neq 0$. Strengthening the constraint would have led to making the DP versus KBL comparison less fair, and most of all: If we are interested in initializing with tighter constraints, we should initialize using the Xia-Yuan linearization constraints.

When running an experiment, the user can set the following the settings for DP: the setting `callback_at` can be set to `DPS.ALL_MIPSOLS` or `DPS.ALL_MIPNODES`, the setting `bd_constr_type` can be set to `USER_CUT` or `DPS.LAZY_CONSTR`, and the setting `minimum_w_difference` can be set to any non-negative floating point number.

The callback function was customized to trigger the separation algorithm at all MILP solutions when `callback_at == DPS.ALL_MIPSOLS`, and to trigger the separation algorithm at branch and cut nodes when `callback_at == DPS.ALL_MIPNODES`. If the separation algorithm is triggered for solution $(\hat{x}, \hat{w})$, then for each (i, j) pair in $[n] \times [n]$ a sub-problem is created. The sub-problem is modeled as an LP, as defined by System (21). Gurobi is used to solve the LP and provide the objective value $\bar{w}_{i,j}$ and the dual multipliers of the sub-problem. If $\hat{w}_{i,j} < \bar{w}_{i,j} -$ `minimum_w_difference`, then constraint (23) is added to the main MILP. This constraint is added in the form of a lazy constraint, using the Gurobipy function `cbLazy()`, if `bd_constr_type == DPS.LAZY_CONSTR`. This constraint is added in the form of a user cut, using the Gurobipy function `cbCut()`, if `bd_constr_type == DPS.USER_CUT`.

Before optimizing, some Gurobi optimization parameters are set. This includes setting the parameter `LazyConstraints` to 1 if `bd_constr_type == DPS.LAZY_CONSTR`, which is required when using lazy constraints. More on the other settings in the next paragraph. Lastly, The main MILP was solved using Gurobi while providing the callback function.

Settings Both KBL and DP have the settings: `pre_crush`, `time_limit`, `threads`, and `soft_mem_limit`. When `pre_crush` is set to `True` the Gurobi parameter `PreCrush` is set to 1. When `PreCrush` is set to 1 Gurobi will allow presolve (Gurobi's solving preparation procedure) to transform any constraint on the original model into an equivalent constraint on the presolved model. This is recommended when using callbacks. For the sake of fair comparison, we set `pre_crush` to `True` for both KBL and DP, when doing experiments. For more information on `PreCrush` see the PreCrush web page. Settings `time_limit`, `threads`, and `soft_mem_limit` are used to set the Gurobi parameters `TimeLimit`, `Threads`, and `SoftMemLimit` respectively.

Measurements Many of the measurements are taken using the Gurobi model attribute. One can for instance call the model attribute `Runtime` after optimizing in order to retrieve the runtime until termination. This runtime excludes the time it took Gurobi to load the model and prepare (presolve) the model. There is no built-in model attribute to retrieve the time spent in the callback. Therefore, we implemented this ourselves inside of DP. We refer to the time spent inside callback as *time inside CB*. The model attribute Runtime includes the time spent in callbacks. In order to be able to compare the time spent outside the callback when using DP, against the runtime when using KBL, we define *time outside CB* to be the model attribute `Runtime` minus the time inside CB.

The model attribute `Work` is a deterministic unit provided by Gurobi. Work quantifies how much time the solving process takes. As mentioned on the Work web page: "Work is deterministic, meaning that you will get exactly the same result every time provided you solve the same model on the same hardware with the same parameter and attribute settings." The model attribute `Work` does not include the `Work` from inside the callback.

Code For more details on the implementation of the method, the experiments and the source code, see github.com/david-van-der-linden/qap-thesis.

4.2 Instances

All instances were taken from the QAPLIB, a Quadratic Assignment Problem Library by R.E. Burkard, S.E. Karisch and F. Rendl. The library consists of 136 different instances, with instance sizes ranging from $n = 10$ till $n = 256$. Some instances are yet to be solved to optimality, while others come with a provided solution. Some instances come from real world applications, while other instances are user generated. All data is provided to fit the Koopmans and Beckmann Problem (KBP) formulation as described in the beginning of Section 1.2. For further reading on the applications of the instances, structure of the data, best known solutions and bounds of instances, we refer to the QAPLIB web page.

4.3 Preliminary Insights

In this subsection, we discuss some preliminary insights that influenced the setup of the main experiment.

Preliminary testing was done on a Lenovo ThinkPad P1, with Processor Intel(R) Core(TM) i7-9750H CPU @ 2.60GH with 16,0 GB of installed RAM. The system type is A windows 64-bit operating system, with a x64-based processor. With Gurobi Optimizer version 9.5.0 build v9.5.0rc5 (win64) installed.

Most preliminary testing was done on the chr18b instance, since this instance is one of the smallest instances that has a solution file available. When running with the settings `callback_at` set to `DPS.ALL_MIPSOLS`, `bd_constr_type` set to `DPS.LAZY_CONSTR` and `minimum_w_difference` set to 0. We noticed that the `Work` of DP (6.08) was less than that of KBL (22.00). See Figure 4 for the results of a recreation of an initial experiment.

Considering that the time inside CB could be reduced a lot, see Section 3, comparing time outside CB of DP against runtime of KBL is interesting. These initial experiments indicate that DP has the potential to be faster than KBL. This begged the question whether the time outside CB of DP also smaller than the runtime of KBL for other instances. These results, with even more information, are available on the github page.

Note that initializing the main MILP with the Kaufman-Broeckx constraints and setting `callback_at` to `DPS.ALL_MIPSOLS` is a slightly strange combination. Since the Kaufman-Broeckx constraints theoretically ensure that w satisfies (9) for all MILP solutions. However, due to numerical in-precision, the difference between $\hat{w}$ and $\bar{w}$ is sometimes slightly more than zero. This will cause the separation algorithm to get called when `minimum_w_difference` set to 0. Since the Benders' cuts were being added and the cuts seemed to be beneficial, we decided to set up the main experiment with this slightly strange combination of initial constraints and `callback_at` setting anyway.

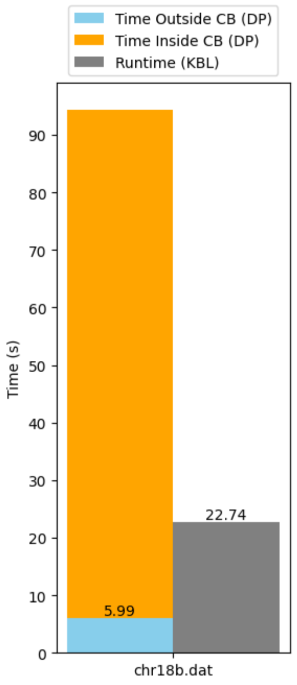


FIGURE 4: Preliminary insights

4.4 Main Experiment

In this subsection, we present the results from the main experiment.

Set Up The main experiment was run on a university owned server named MORDOR. The server has the following specifications: CPU model: Intel(R) Xeon(R) Gold 5217 CPU @3.00GHz, 16 physical cores, 32 logical processors, 64 GB Memory. With Gurobi optimizer version 10.0.1 build v10.0.1rc0 (linux64) installed. Gurobi settings specified one thread was allowed to be used for optimization, the experiments were run with a time limit of 1 hour per instance, and a soft memory limit of 3.6 GB. DP was run with the settings `callback_at` set to `DPS.ALL_MIPSOLS`, `bd_constr_type`, set to `DPS.LAZY_CONSTR` and `minimum_w_difference` set to 0. Experiments were run with up to 10 instances running in parallel.

Results When running the main experiment on 19 of the instances, the program halted with the error: Unable to retrieve attribute 'X'. We suspect that this error occurred since we try to read off the best known solution, while no solution is found. To be precise this occurred on when running instances: sko100d, lipa90a, sko100e, lipa90b, sko100f, sko100a, sko100b, sko100c, tai100a, tho150, tho30, tho40, wil100, tai150b, tai256c, tai25a, tai25b, tai30a, tai30b.

From here on out, we will only consider the results of the remaining 117 instances. As expected, not all instances were solved to optimality, see Table 1 and Figure 5. See Figure 6 for a more detailed comparison. Unsurprisingly, on most instances the time limit was reached, and some instances triggered the memory limit. From Figure 6 we conclude: no instance that was solved with DP that did not get solved with KBL. This can be explained by the long time outside CB.

When considering the instances that were solved to optimality using both methods, plotting the time yields Figure 7. The KBL Runtime is smaller than the time outside CB for DP on most instances. Therefore, the figure does not indicate DP has potential to be

Model Status	DP	KBL
Optimal	19	26
Timeout	86	70
MemLimit	12	21

Table 1: Main experiment: Number of instances per model status.

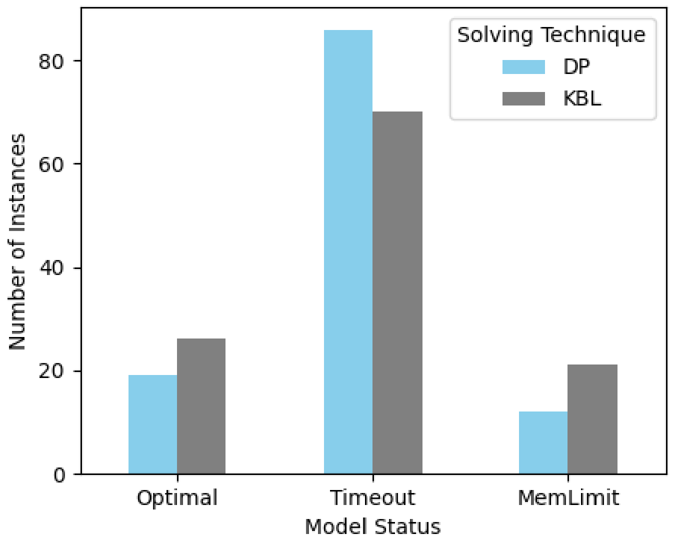


Figure 5: Main experiment: Number of instances per model status.

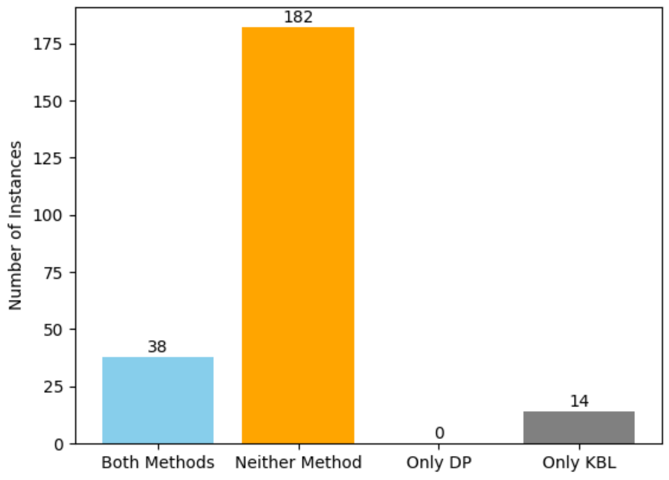


Figure 6: Main experiment: Solved to optimality with both methods, neither method, only DP, or only KBL

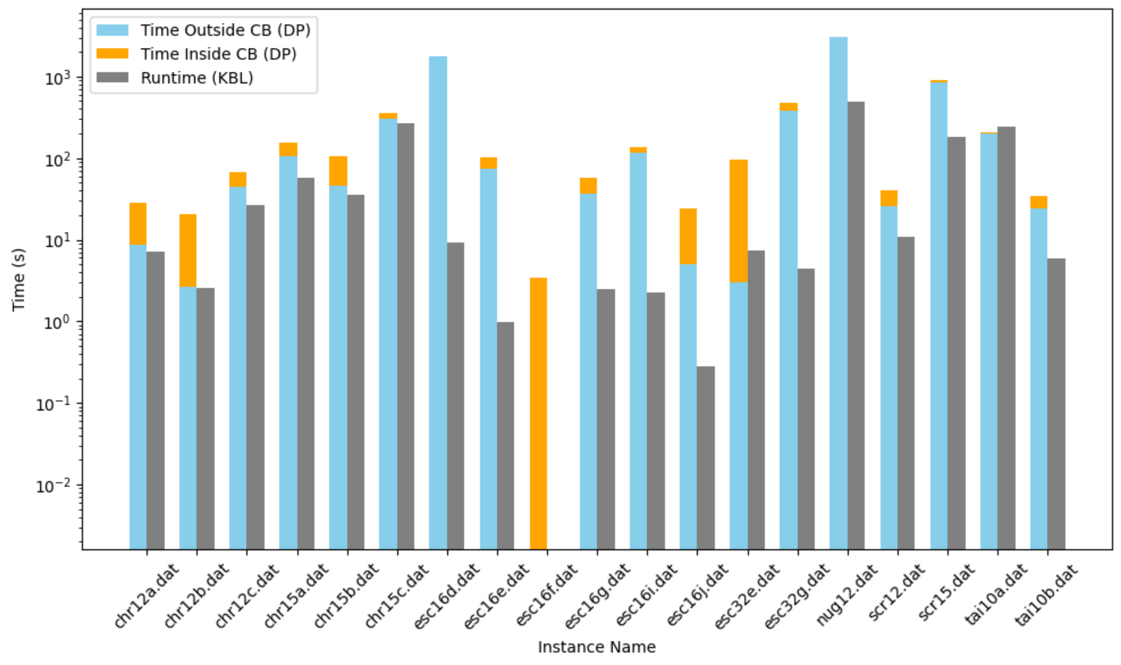


Figure 7: Main experiment: Time per instance for all instances that both methods solved to optimality. Note that the time axis is logarithmic. For Instance esc16f.dat the runtime (KBL) and time outside CB (DP) are less than one micro second, while the time inside CB is a matter of seconds.

faster than KBL. However, upon closer inspection, we notice that when running DP on chr18b the time limit is reached, while when running KBL on chr18b an optimal solution is found within a runtime of 20.3 seconds. This is unexpected since we expect runtimes to be lower, or at least similar, when running on MORDOR than when running on a Lenovo ThinkPad P1, however that does not seem to be the case for DP.

4.5 Later Results

Later trails with the Gurobi version of MORDOR set to Gurobi Optimizer version 9.5.1 build v9.5.1rc2 (linux64), again resulted in the time-out limit being reached when running the DP solving technique on instance chr18b.

5 Discussion and Conclusion

Investigating the discrepancy between results when running DP on chr18b on the Lenovo ThinkPad P1 and MORDOR requires more time. This should however be considered, as currently no conclusions on the utility of the DP method over the KBL method can be drawn from the computational experiments.

5.1 Recommendations Further Experiments and Development of DP

User Cuts at MILP Nodes For further experiments, we recommend researching callbacks at MILP nodes. However, as mentioned by in the Gurobi documentation, user cuts should be added sparingly, since they increase the size of the relaxation model that is solved at each node and can significantly degrade node processing speed. We recommend initially investigating the increase in strength of the model if all cuts were added in order to identify the potential. If the bound improvement is significant, we recommend investigating heuristics to identify which cuts are best to add and which cuts are best to be left out, and investigate whether it's beneficial to add cuts more sparingly.

Numerical Focus Parameter During the main experiment, Gurobi gave the following two warnings and advice when running on some of the instances:

> "Warning: Model contains large matrix coefficients
>
> Warning: Model contains large rhs
>
> Consider reformulating model or setting the NumericalFocus parameter to avoid numerical issues"

When inspecting the data, we noticed some of the instances do contain large coefficients. Note that one of the instances has a flow entry of over 5000, which if it matches with a distance entry of similar size yields a q entry of over 5000^2, while the model will also have constraints which coefficients of 1, making scaling a nontrivial task. We recommend testing out various settings for the Numerical Focus parameter, and look into reformulating the model using auxiliary variables that are scaled versions of the original variables.

Different Server The server MORDOR is a general purpose Linux server. It has no job scheduling or any other features to ensure accurate and reliable performance testing. When doing future experiments that require accurate runtime measurements, we recommend using a server dedicated for performance testing.

Xia-Yuan Linearization Since the Xia-Yuan linearization (XY), see Section 1.3.3, seems to be similar to and stronger than the Kaufman-Broeckx linearization, comparing the performance of DP versus an Xia-Yuan linearization implementation is a logical next step if the DP compares well against KBL.

Initial constraints The computational experiments in this thesis were performed while the DP model was initialized using the KBL constraints. One could investigate the effect of initializing the DP model with the XY constraints. And one could investigate what the constraints are that are added when not initializing KBL nor XY. This would make it that there are no initial constraints that ensure that w and x relate as desired for integer points. This could result in the callback adding other constraints.

Investigate Bounds RLT The Reformulation Linearization Technique (RLT) is a technique that can be applied to some linearization of the QAP. A rough draft of such a linearization has been implemented and can be found on github.com/david-van-der-linden/qap-thesis. We are interested in seeing how the bounds of RLT model compare to that of the KBL and the DP model. This could be used to judge the strength of the DP model.

Speeding up Lifting LP As before mentioned in Section 3: If the Disjunctive Programming method seems promising based on other results, we recommend proving the reduction to minimum cost flow problem and implementing primal-dual algorithm in a programming language that has low runtime, for example Rust, C, or C++.

Symmetry For further improvement of the proposed method, we recommend making use of symmetry. The use of symmetry, as mentioned by Fischetti et al. [10], plays a large role in solving some instances. The symmetry $q_{i,j,k,l} = q_{k,l,i,j}$ and other symmetries are underutilized in the current implementation of DP.

References

[1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows: theory, algorithms, and applications.* Prentice Hall, Upper Saddle River, NJ, 1993.

[2] Egon Balas. Disjunctive Programming: Cutting Planes from Logical Conditions. In O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, editors, *Nonlinear Programming 2*, pages 279–312. Academic Press, January 1975. `doi:10.1016/B978-0-12-468650-2.50015-8`.

[3] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, December 1962. `doi:10.1007/BF01386316`.

[4] R. E. Burkard and J. Offermann. Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme. *Zeitschrift für Operations Research*, 21(4):B121–B132, August 1977. `doi:10.1007/BF01918175`.

[5] Rainer E. burkard. Quadratic Assignment Problems. In *Handbook of Combinatorial Optimization*, pages 2741–2814. Springer New York, 2013. `doi:10.1007/978-1-4419-7997-1_22`.

[6] Rainer E. Burkard, Eranda Çela, Panos M. Pardalos, and Leonidas S. Pitsoulis. The Quadratic Assignment Problem. In *Handbook of Combinatorial Optimization*, pages 1713–1809. Kluwer Academic Publishers, 1998. `doi:10.1007/978-1-4613-0303-9_27`.

[7] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer Programming*, volume 271 of *Graduate Texts in Mathematics*. Springer International Publishing, Cham, 2014. `doi:10.1007/978-3-319-11008-0`.

[8] J. W. Dickey and J. W. Hopkins. Campus building arrangement using topaz. *Transportation Research*, 6(1):59–68, March 1972. `doi:10.1016/0041-1647(72)90111-6`.

[9] Julius Farkas. Theorie der einfachen Ungleichungen. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1902(124):1–27, January 1902. Publisher: De Gruyter Section: Journal für die reine und angewandte Mathematik. `doi:10.1515/crll.1902.124.1`.

[10] Matteo Fischetti, Michele Monaci, and Domenico Salvagnin. Three Ideas for the Quadratic Assignment Problem. *Operations Research*, 60(4):954–964, 2012. `doi:10.1287/opre.1120.1073`.

[11] R. L. Francis, Leon Franklin McGinnis, and John A. White. *Facility Layout and Location: An Analytical Approach.* Prentice Hall, 1992. Google-Books-ID: eS4fAQAAIAAJ.

[12] David Gale, Harold Kuhn, and Albert W. Tucker. Linear programming and the theory of games. In *Activity analysis of production and allocation*, volume 13, pages 317–335. Cowles Commission Monographs, 1951. URL: `https://www.ma.imperial.ac.uk/~ajacquie/IC_Num_Methods/IC_Num_Methods_Docs/Literature/Gale.pdf`.

[13] P. C. Gilmore. Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem. *Journal of the Society for Industrial and Applied Mathematics*, 10(2):305–313, 1962. `doi:10.1137/0110022`.

[14] L. Kaufman and F. Broeckx. An algorithm for the quadratic assignment problem using Benders' decomposition. *European Journal of Operational Research*, 2(3):207–211, 1978. `doi:10.1016/0377-2217(78)90095-4`.

[15] Tjalling C. Koopmans and Martin Beckmann. Assignment Problems and the Location of Economic Activities. *Econometrica*, 25(1):53–76, 1957. Publisher: [Wiley, Econometric Society]. `doi:10.2307/1907742`.

[16] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. `doi:10.1002/nav.3800020109`.

[17] Eugene L. Lawler. The Quadratic Assignment Problem. *Management Science*, 9(4):586–599, 1963. `doi:10.1287/mnsc.9.4.586`.

[18] Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, and Tania Querido. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2):657–690, January 2007. `doi:10.1016/j.ejor.2005.09.032`.

[19] Jiří Matoušek and Bernd Gärtner. *Understanding and Using Linear Programming*. Universitext. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. `doi:10.1007/978-3-540-30717-4`.

[20] G. Miranda, H. P. L. Luna, G. R. Mateus, and R. P. M. Ferreira. A performance guarantee heuristic for electronic components placement problems including thermal effects. *Computers & Operations Research*, 32(11):2937–2957, November 2005. `doi:10.1016/j.cor.2004.04.014`.

[21] NikNaks. Rook, April 2013. URL: `https://commons.wikimedia.org/wiki/File:Chess_rlt26.svg#Licensing`.

[22] Chris Oakman. chessboardjs. URL: `https://chessboardjs.com`.

[23] Maurice Queyranne. Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems. *Operations Research Letters*, 4(5):231–234, February 1986. `doi:10.1016/0167-6377(86)90007-6`.

[24] Sartaj Sahni and Teofilo Gonzalez. P-Complete Approximation Problems. *Journal of the ACM*, 23(3):555–565, July 1976. `doi:10.1145/321958.321975`.

[25] Leon Steinberg. The Backboard Wiring Problem: A Placement Algorithm. *SIAM Review*, 3(1):37–50, 1961. Publisher: Society for Industrial and Applied Mathematics. URL: `https://www.jstor.org/stable/2027247`.

[26] Yong Xia. Gilmore-Lawler bound of quadratic assignment problem. *Frontiers of Mathematics in China*, 3(1):109–118, February 2008. `doi:10.1007/s11464-008-0010-4`.

[27] Yong Xia and Ya-Xiang Yuan. A new linearization method for quadratic assignment problems. *Optimization Methods and Software*, 21(5):805–818, October 2006. `doi:10.1080/10556780500273077`.

[28] Huizhen Zhang, Cesar Beltran-Royo, and Liang Ma. Solving the quadratic assignment problem by means of general purpose mixed integer linear programming solvers. *Annals of Operations Research*, 207(1):261–278, 2012. `doi:10.1007/s10479-012-1079-4`.

During the preparation of this work, the authors used LanguageTool in order to correct spelling and grammar. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the work.