Comparing Solution Methods for the Quadratic Bin Packing Problem

January 2025

A Thesis for the Bachelor Applied Mathematics
at the TU Delft

By
Hendrik de Mol van Otterloo
Student Number 4736796

Supervised by
Frank de Meijer

Assessed by
Wolter Groenevelt
Frank de Meijer

To be defended on
Thursday 23, January 2025 at 10:00 o'clock

Contents

Abstract

The Bin Packing Problem is a famous $\mathcal{NP}$-hard problem in the field of optimization where items need to be packed in as few bins as possible. The problem has many applications across different fields. The Quadratic Bin Packing Problem (QBPP) is an extension of the Bin Packing Problem where a joint cost is added when two items are packed together and is $\mathcal{NP}$-hard as well. The problem is quite unknown but has applications is for example hospital organisations or advertisement placement. In this thesis we relax the QBPP via linear programming and semidefinite programming and compare these relaxations by relative optimality gap and computation time. In particular the SDP relaxation has not been done before and has been found to generally work well in relaxing quadratic problems. We find that for low values of n the problem can best be solved as an integer quadratic program. For large values of n we find the SDP relaxation to be optimal giving tight bounds in a reasonable amount of time. Like similar research in this field, we also find SDP relaxations give a strong and fast bound to integer quadratic programs.

1 Introduction

The Bin Packing Problem (BPP) is a widely known and studied problem across the field of mathematics. The problem has many applications such as in healthcare or logistics. For more examples, Munien and Ezugwu [25] have compiled some of the many applications of the BPP. The objective of the BPP is to pack a number of items with different weights in as few bins as possible. Every bin has a capacity and the items in a bin may not exceed the capacity of the bin. The objective is to minimize the number of bins used to pack all items. For a general introduction to the Bin Packing Problem, we recommend the book by Scheithauer [31].

The BPP has a quadratic extension known as the Quadratic Bin Packing Problem (QBPP). In this problem a pair of items packed together also provide additional costs or reduction in costs. The exact problem has not been studied much, and to the best of our knowledge the problem was first formally introduced in the thesis by Chagas [9] in 2021. The class of general 0-1 quadratic problems, of which the QBPP is one, has been studied in, for example, this article by Caprara [8].

An example of an application of the QBPP is to minimize the number of servers required to process a given set of tasks under server capacity constraints as described by Aydin et al [1]. In this example the servers are the bins and the tasks are the items that need to be handled within the capacity of the servers. In a quadratic extension certain tasks on the same server could be more expensive.

Another example can be found in hospital planning, as inspired by an article by Laurent and Klement [22]. We have certain classes of operations that need to be packed across hospitals without exceeding capacity, where certain operations organised together in the same hospital save costs by using the same equipment for multiple operations.

The QBPP is $\mathcal{NP}$-hard as is explained in Chapter 2, which makes finding an optimal solution hard, hence solutions are often found using a relaxation and a branch-and-bound technique. Applying a branch-and-bound technique is expensive which is why it is relevant to find a tight bound on the relaxation. The branch-and-bound technique is beyond the scope of this thesis, though Chagas [9] has done some experiments with applying different branch-and-bound techniques to the QBPP if the reader desires to learn more.

In this thesis we will compare relaxations of the QBPP: the relaxation of the linearization and the relaxation as a semidefinite problem (SDP). In particular the SDP relaxation has not yet been applied to this problem and we find reason to believe this method may yield strong results, as we will explain in Chapter 4. We will compare the relaxations based on the tightness of the bound and the computation time.

This thesis is organised as follows: first, we give an introduction to the QBPP in Chapter 2. We relax the QBPP to its linearized form in Chapter 3 and its SDP form in Chapter 4. We will describe the methods used to compare these relaxations in Chapter 5, describe the results in Chapter 6 and conclude the thesis in Chapter 7.

1.1 Notation

We will use $[n]$ to denote the set of integers from 1 to n. The trace inner product is a mapping $\mathbb{R}^{n\times m} \times \mathbb{R}^{n\times m} \to \mathbb{R}$ such that $C \bullet X = \sum_{i=1}^{n} \sum_{j=1}^{m} X_{ij} C_{ij}$. We denote the set of $n \times n$ symmetric real matrices as $\mathbb{S}^n$. We denote with diag: $\mathbb{R}^{n\times n} \to \mathbb{R}^n$ the operator that maps a matrix to a vector containing the values on the diagonal. We use Diag: $\mathbb{R}^n \to \mathbb{R}^{n\times n}$ to create a matrix with the input vector on the diagonal and 0 elsewhere. We use rank: $\mathbb{R}^{n\times n} \to \mathbb{N}$ to denote the rank of a matrix. The matrix J denotes the matrix with every element equal to 1.

2 The Quadratic Bin Packing Problem

In this chapter we give an introduction to the QBPP. We first describe the BPP and its related works, then extend the BPP to the QBPP. We also give an example of a QBPP and discuss some related works.

2.1 The Bin Packing Problem

The Bin Packing Problem is a problem where a certain number of items need to be packed in as few bins as possible. Each bin has a certain capacity, and all items a certain weight. The weight of the items packed in a bin must not exceed the capacity. Thus, an instance of the BPP requires a finite set $[n]$ that represents any n number of items, a function $s : [n] \to \mathbb{R}$ that assigns to each item a weight, and a parameter $W \in \mathbb{R}$ that determines the capacity of all bins. The objective of the problem is to minimize the amount of bins used. We use variable $x_i^k \in \{0,1\}$ to denote if item i is in bin k:

$$x_i^k = \begin{cases} 1, & \text{if item } i \text{ is in bin } k, \\ 0, & \text{otherwise.} \end{cases}$$

We assume in the BPP that every item can be packed, or alternatively, that the weight of each item is smaller than the capacity of a bin, otherwise there is an item that cannot be packed and the problem is infeasible. It is then obvious to see that a feasible solution to the BPP is to pack each item in its own bin. Thus any solution to the BPP uses at most n bins.

We use binary variable y_k to denote whether there is at least one item in bin k, such that

$$y_k = \begin{cases} 1, & \text{if there is an item in bin } k, \\ 0, & \text{otherwise.} \end{cases}$$

This results in the following integer linear program:

$$\begin{aligned}
\min \quad & \sum_{k\in[n]} y_k & & (1)\\
\text{s.t.} \quad & x_i^k \le y_k & \forall i,k \in [n], \quad & (2)\\
& \sum_{k\in[n]} x_i^k = 1 & \forall i \in [n], \quad & (3)\\
& \sum_{i\in[n]} w_i x_i^k \le W y_k & \forall k \in [n], \quad & (4)\\
& x_i^k, y_k \in \{0,1\} & \forall i,k \in [n]. \quad & (5)
\end{aligned}$$

The objective function sums over all bins that are used, which we want to minimize. Constraints (2) then force a bin to be used if there is at least one item in it. Constraints (3) force all items to be in exactly one bin while constraints (4) force the bins not to exceed their weight limit. Finally, constraints (5) force the decision variables x_i^k and y_k to be binary.

The BPP, also referred to as the one-dimensional Cutting Stock Problem or the one-dimensional Bin Packing Problem, has been studied extensively for many years, even as far back as 1979[5]. The problem is known to be $\mathcal{NP}$-hard[16]. Though the BPP has been widely studied, new advances are still being made. For example, a survey by Munien and Ezugwu [25] explores recent advances made in metaheuristics for the BPP. A article by Delorme et al. [13] reviews some of the current optimal ways of solving the BPP. For a general introduction to the Bin Backing Problem we once more recommend the book by Scheithauer [31], who explores lower bounds, heuristics, exact approaches and algorithms for the BPP. The book by Korte and Vygen [21] also describes the BPP while being more focused on algorithms, heuristics and approximation schemes to the problem.

Common extensions to the BPP are multi-dimensional bin packing problems where the items and bins have multiple dimensions, and the capacity constraints need to hold in all dimensions. Most common are the 2-dimensional and 3-dimensional Bin Packing Problem, with applications in for example computer storage, where files of different sizes have to be stored [18]. The problem is more complex than the BPP, making the multi-dimensional bin packing problem $\mathcal{NP}$-hard. A survey of recent advances in the general class of multi-dimensional bin packing problems has recently been done by Christensen et al. [11].

2.2 Quadratic Bin Packing Problem

The QBBP is an extension of the BPP and introduces a new function $d : [n] \times [n] \to \mathbb{R}$ representing dissimilarity. This is an extra or reduced cost when two items are packed in the same bin. We assume $d_{ij} = d_{ji}$, so the function is symmetric, thus the order in which items are packed is irrelevant. We also assume, without loss of generality, $d_{ii} = 0,\ \forall\, i \in [n]$, so an item does not incur extra costs when packed with itself. If $d_{ii} \neq 0$, then, because every item is packed exactly once, this cost is a constant unavoidable extra cost which is trivial for a minimization problem.

The dissimilarity function changes the objective function to minimize costs, as opposed to minimizing bins, and requires a cost for using a bin. We use parameter α to denote the cost of using a bin.

We can formulate the QBPP as an integer quadratic program as follows:

$$
\begin{array}{llcrr}
F_{IQP} & \min & \frac{1}{2}\sum_{i\in[n]}\sum_{j\in[n]}\sum_{k\in[n]} d_{ij}x_i^k x_j^k + \alpha \sum_{k\in[n]} y_k & & (6)\\
 & \text{s.t.} & x_i^k \leq y_k & \forall i,k \in [n], & (7)\\
 & & \sum_{k\in[n]} x_i^k = 1 & \forall i \in [n], & (8)\\
 & & \sum_{i\in[n]} w_i x_i^k \leq W y_k & \forall k \in [n], & (9)\\
 & & x_i^k, y_k \in \{0,1\} & \forall i,k \in [n]. & (10)
\end{array}
$$

We will refer to this program as the F_{IQP} formulation.

Constraints (7) - (10) are equal to (2) - (5). The objective function (6) minimizes costs. Because we sum over all pairs of (i, j) in the objective function, we add all dissimilarity values twice. For example, if items (2,3) are together in the same bin, we add the dissimilarity once for the pair (2,3) and once again for pair (3,2). To correct this we multiply the first part of our objective function by $\frac{1}{2}$.

The objective function contains the product of x_i^k and x_j^k, which is 1 if item i and j are together in bin k and 0 otherwise. We multiply this with the dissimilarity value d_{ij} to get the value of the reduced or extra cost. The objective function also adds the cost of a bin, α, for each bin used.

Note how the solution to the problem is not unique. Because the ordering of the bins is arbitrary, we can shuffle the numbering of the used bins and the optimal value will not change. we will expand on this in Section 2.4.

The product of the decision variables x_i^k and x_j^k makes the problem quadratic, giving the problem its name. If we take $d_{ij} = 0,\ \forall (i, j) \in [n]$ we reduce the problem to the BPP. Thus, because the QBPP adds complexity to the BPP, which is $\mathcal{NP}$-hard [16], the QBPP is $\mathcal{NP}$-hard as well.

2.2.1 Example QBPP

In this Section we give an example of a QBPP. Suppose we have 5 items such that:

item	1	2	3	4	5
weight	1	6	2	4	5

The capacity of each bin is set to 10. If this were a Bin Packing Problem this problem would be easy to solve, as it is easy to see that all items can be packed in 2 bins. For example packing item 1, 2 and 3 in a bin, and packing item 4 and 5 together.

For the QBPP we add a cost for each bin, which we set to 6. We also add a dissimilarity matrix:

$$
D = \begin{bmatrix} 0 & 5 & -1 & 2 & 0 \\ 5 & 0 & 5 & 4 & 3 \\ -1 & 5 & 0 & 2 & 5 \\ 2 & 4 & 2 & 0 & -1 \\ 0 & 3 & 5 & -1 & 0 \end{bmatrix}
$$

such that $d_{ij} = D_{ij}$. Note how the optimal solution is much harder to intuit than for the BPP. We get an optimum as in Figure 1 with optimal value 16. The total cost deriving from using the 3 bins costing 18 and the dissimilarities saving 2. It turns out packing item 1 and 3 separately is cheaper than packing them with item 2 due to the dissimilarity costs.

Figure 1: Solution to the example QBPP

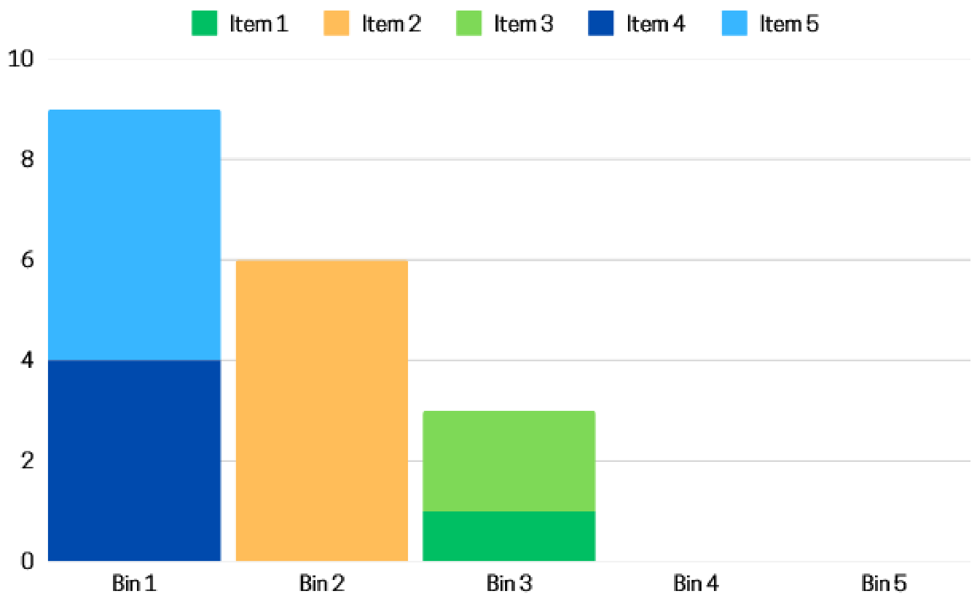
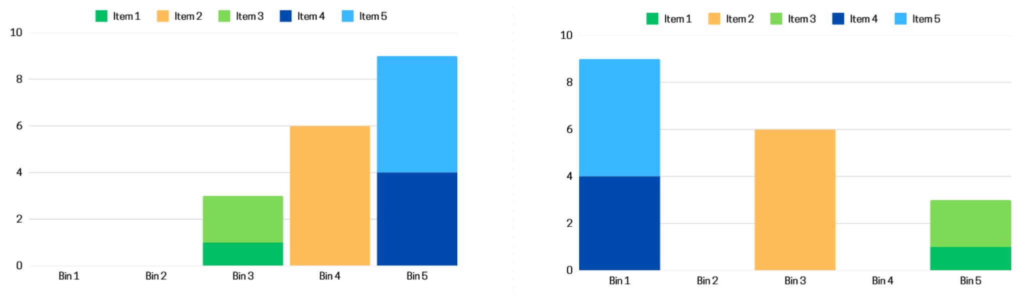


Figure 2: Different but optimal solutions to the example QBPP



2.2.2 Related Problems and Works

A problem closely related to the QBPP is the Quadratic Knapsack Problem (QKP), first introduced by Gallo et al. [15]. It is similar to the QBPP except the objective is to maximize profits as opposed to minimize costs, where each item carries a profit instead of a cost. There is only one bin and not all items have to be packed. Similar to the QBPP, items packed together provide a joint cost. Pisinger [27] has studied various upper bounds of the relaxation of this problem. There is also a study of various algorithms for solutions of the QKP by Pardalos et al. [26]. For a more general introduction we recommend the book by Gallo et al. [15].

Another problem closely related to the QBPP is the Quadratic Multiple Knapsack Problem (QMKP). It is a relatively new problem and was first introduced by Hiley and Julstrum [19] in 2006. It is similar to the QKP, with multiple knapsacks of varying capacities. A lot of advances are still being made on this problem. See for example the study by Bergman [3] on exact solution methods for solving the QMKP, or the article by Saraç and Sipahioglu [30] who studied different algorithms for solving the QMKP.

For both of the problems stated above, an overview of recent advances in the field of knapsack problems was recently done by Cacchiani et al. [7].

2.3 Relaxation of the QBPP

To solve the QBPP we relax the problem from a mixed integer quadratic program to a quadratic program. We then solve this quadratic program to get a lower bound of the optimal solution and can then find a solution to the integer quadratic problem using a branch-and-bound technique.

The specifics of the branch-and-bound technique is beyond the scope of this thesis, though if the reader desires to read more, we recommend this book by Brusco and Stahl [6] who provide a good examination to the branch and bound technique. Also of interest is this recent survey by Huang et al. [20] who analyse the recent advances in branch and bound techniques using machine learning. For a reference more closely related to the QBPP, the reader can also read this article by Sadykov and Vanderbeck [29] who apply a branch-and-price algorithm, which is similar to the branch-and-bound method, to the Bin packing problem. We reiterate that the relevance of the branch-and-bound method for this thesis is that it is computationally expensive to solve, thus a relaxation that can give an optimal value close to the optimal solution of the QBPP is very much desired.

To relax the F_{IQP} formulation of the QBPP we drop constraints (10) and bound x and y to be between 0 and 1 by adding the following constraint:

$$x_i^k \geq 0 \qquad \forall\, i, k \in [n], \tag{11}$$
$$y^k \leq 1 \qquad \forall\, k \in [n]. \tag{12}$$

Note that the above constraints with (7) bound all variables between 1 and 0. We will refer to the relaxation of the QBPP as the F_{QP} formulation. We will state it once more below for completeness.

$$\begin{aligned}
F_{QP} \qquad \min \qquad & \frac{1}{2}\sum_{i\in[n]}\sum_{j\in[n]}\sum_{k\in[n]} d_{ij}x_i^k x_j^k + \alpha \sum_{k\in[n]} y_k && && (13)\\
\text{s.t.} \qquad & x_i^k \leq y_k && \forall i, k \in [n], && (14)\\
& \sum_{k\in[n]} x_i^k = 1 && \forall i \in [n], && (15)\\
& \sum_{i\in[n]} w_i x_i^k \leq W y_k && \forall k \in [n], && (16)\\
& x_i^k \geq 0 && \forall i, k \in [n], && (17)\\
& y_k \leq 1 && \forall k \in [n]. && (18)
\end{aligned}$$

2.4 Symmetry Reduction

Because the bins are all identical and their ordering is arbitrary, our current problem has multiple optimal solutions. Let r be the number of bins our optimal solution uses, then we have exactly $n!/(n-r)!$ combinations and optimal

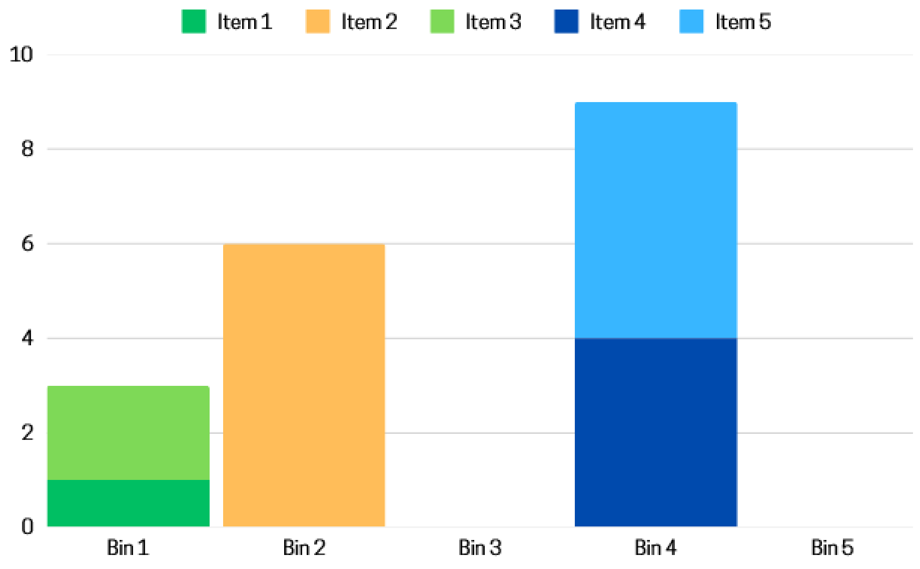


Figure 3: The optimal solution when reduced by constraints (19) and (20)

solutions. For larger n this number increases incredibly fast. Because a problem with a single solution might be computationally easier to solve, as is the case for many integer linear programs [24], we can add more constraints around the index numbers of the bins to reduce the problem to a single optimal solution. One way to do this was found by Bonami et al. [4].

We add the following constraints:

$$x_i^k = 0 \qquad \forall i, k \in [n], k > i, \tag{19}$$

$$x_j^i \leq x_i^i \qquad \forall i, j \in [n], j > i. \tag{20}$$

Constraints (19) only allow item i to be in bin i or a bin of a lower index number. Constraints (20) puts a restriction on the bin number as follows:

$$\text{bin } k = \begin{cases} \text{is empty,} \\ \text{contains item } k. \end{cases}$$

Thus every pack of items is always in the bin of the item with the lowest index number. Referring to the example once again, we would get the single optimal solution as can be seen in Figure 3.

When we add these constraints to F_{QP} we see that we can replace the variable y_k with x_k^k. Because $x_k^k = 1$ only if a bin is used, these variables fulfil the same function.

We can then reformulate F_{QP} with symmetry reduction as follows:

$$
\begin{aligned}
F_{QP}^{+} \qquad \min \quad & \frac{1}{2}\sum_{i\in[n]}\sum_{j\in[n]}\sum_{k\in[n]} d_{ij}x_i^k x_j^k + \alpha \sum_{k\in[n]} x_k^k && && (21)\\
\text{s.t.} \quad & x_i^k \le x_k^k && \forall i,k \in [n], && (22)\\
& \sum_{k\in[n]} x_i^k = 1 && \forall i \in [n], && (23)\\
& \sum_{i\in[n]} w_i x_i^k \le W y_k && \forall k \in [n], && (24)\\
& x_i^k = 0 && \forall i,k \in [n], \quad k > i, && (25)\\
& 0 \le x_i^k \le 1 && \forall i,k \in [n]. && (26)
\end{aligned}
$$

We will refer to this relaxation as F_{QP}^{+}.

We can also apply symmetry reduction to the F_{IQP} method to get the F_{IQP}^{+} formulation as follows:

$$
\begin{aligned}
F_{IQP}^{+} \qquad \min \quad & \frac{1}{2}\sum_{i\in[n]}\sum_{j\in[n]}\sum_{k\in[n]} d_{ij}x_i^k x_j^k + \alpha \sum_{k\in[n]} x_k^k && && (27)\\
\text{s.t.} \quad & x_i^k \le x_k^k && \forall i,k \in [n], && (28)\\
& \sum_{k\in[n]} x_i^k = 1 && \forall i \in [n], && (29)\\
& \sum_{i\in[n]} w_i x_i^k \le W y_k && \forall k \in [n], && (30)\\
& x_i^k = 0 && \forall i,k \in [n], \quad k > i \quad , && (31)\\
& x_i^k \in \{0,1\} && \forall i,k \in [n]. && (32)
\end{aligned}
$$

We will compare both the F_{IQP} and F_{IQP}^{+} formulations with the relaxations.

Because the symmetry reducing constraints add constraints to our existing program, we can say that any relaxation with symmetry reducing constraints is stronger than its regular counterpart. We can thus expect for a method with symmetry reduction an optimality gap that is at most the gap of its counterpart in the results.

3 Linearizing the QBPP

The relaxation of the QBPP is computationally expensive to solve because of the quadratic component in the objective function. Hence we might improve performance by linearizing the QBPP, and solving a linear program (LP). In this chapter we reformulate the F_{QP} and F_{QP}^{+} relaxations to LPs. We will do so using three different methods:

The method by Glover and Woolsey [17] which is used for general quadratic 0-1 programs. Chagas[9] seems to find the method by Fan and Pardalos[14] most effective for linearizing the QBPP. Lastly we will use an abridged version of the method by Fan and Pardalos that has to the best of our knowledge not yet been applied to this problem.

3.1 Linearization

For an optimal solution of the QBPP all variables are binary. Hence a linearization must be true only when the x_i^k is binary. Thus we can linearize the quadratic objective function as if it were binary without losing the optimal solution. When we linearize a quadratic binary function we can reduce the quadratic equation to 4 cases. Defining $z_{ij}^k = x_i^k \cdot x_j^k$ we get:

x_i^k	x_j^k	z_{ij}^k
0	0	0
0	1	0
1	0	0
1	1	1

Thus we want to add linear constraints to F_{QP} to force z_{ij}^k to take on the correct result depending on the values of x_i^k and x_j^k, so that we can replace the objective function of the QBPP with the following:

$$\min \frac{1}{2} \sum_{i\in[n]} \sum_{j\in[n]} \sum_{k\in[n]} d_{ij} z_{ij}^k + \alpha \sum_{k\in[n]} y_k. \tag{33}$$

Because $z_{ij}^k = x_i^k \cdot x_j^k = x_j^k \cdot x_i^k$ and the ordering of j and i is arbitrary we need the above logic table to only hold for $j \geq i$. For example $x_4^k \cdot x_2^k = x_2^k \cdot x_4^k = z_{2,4}^k$, which makes the table for $j \leq i$ redundant.

3.1.1 Method of Glover and Woolsey

The method of Glover and Woolsey introduces the following constraints:

$$z_{ij}^k \leq x_i^k \qquad \forall\, i,j,k \in [n], \qquad j \geq i, \tag{34}$$

$$z_{ij}^k \leq x_j^k \qquad \forall\, i,j,k \in [n], \qquad j \geq i, \tag{35}$$

$$x_i^k + x_j^k - z_{ij}^k \leq 1, \qquad \forall\, i,j,k \in [n], \qquad j \geq i, \tag{36}$$

$$z_{ij}^k = z_{ji}^k \qquad \forall i,j,k \in [n], \qquad j < i, \tag{37}$$

$$0 \leq z_{ij}^k \leq 1 \qquad \forall\, i,j,k \in [n], \qquad j \geq i. \tag{38}$$

We can see that for every bin $k \in [n]$, $z_{ij}^k = 1$ only if $x_i^k = 1$ and $x_j^k = 1$ by constraints (36). If either $x_i^k = 0$ or $x_j^k = 0$ then constraints (34) and (35) force $z_{ij}^k = 0$. Finally constraints (38) bound z_{ij}^k between 0 and 1 and constraints 37 force the symmetry of z_{ij}^k. If we add constraints forcing x_i^k to be binary then z_{ij}^k must be binary as well.

We can thus add constraints (34) - (38) to F_{QP} and F_{QP}^{+} with objective function (33) and get new relaxations of the QBPP. We will call these methods $F_{G\&W}$ and $F_{G\&W}^{+}$, respectively.

For the $F_{G\&W}$ method we get the following:

$$
\begin{array}{llllll}
F_{G\&W} & \min & \frac{1}{2}\sum_{i\in[n]}\sum_{j\in[n]}\sum_{k\in[n]} d_{ij}z_{ij}^k + \alpha\sum_{k\in[n]} y_k & & & (39)\\
& \text{s.t.} & x_i^k \leq y_k & \forall i,k\in[n], & & (40)\\
& & \sum_{k\in[n]} x_i^k = 1 & \forall i\in[n], & & (41)\\
& & \sum_{i\in[n]} w_i x_i^k \leq W y_k & \forall k\in[n], & & (42)\\
& & z_{ij}^k \leq x_i^k, & \forall\, i,j,k\in[n], & j\geq i, & (43)\\
& & z_{ij}^k \leq x_j^k, & \forall\, i,j,k\in[n], & j\geq i, & (44)\\
& & x_i^k + x_j^k - z_{ij}^k, \leq 1, & \forall\, i,j,k\in[n], & j\geq i, & (45)\\
& & x_i^k \geq 0 & \forall i,k\in[n], & & (46)\\
& & y_k \leq 1 & \forall k\in[n], & & (47)\\
& & z_{ij}^k = z_{ji}^k & \forall i,j,k\in[n], & j<i, & (48)\\
& & 0\leq z_{ij}^k \leq 1, & \forall\, i,j,k\in[n], & j\geq i. & (49)
\end{array}
$$

And for $F^+_{G\&W}$ we get:

$$
\begin{array}{llllll}
F^+_{G\&W} & \min & \frac{1}{2}\sum_{i\in[n]}\sum_{j\in[n]}\sum_{k\in[n]} d_{ij}z_{ij}^k + \alpha\sum_{k\in[n]} x_k^k & & & (50)\\
& \text{s.t.} & x_i^k \leq x_k^k & \forall i,k\in[n], & & (51)\\
& & \sum_{k\in[n]} x_i^k = 1 & \forall i\in[n], & & (52)\\
& & \sum_{i\in[n]} w_i x_i^k \leq W y_k & \forall k\in[n], & & (53)\\
& & x_i^k = 0 & \forall i,k\in[n], & k>i, & (54)\\
& & z_{ij}^k \leq x_i^k, & \forall\, i,j,k\in[n], & j\geq i, & (55)\\
& & z_{ij}^k \leq x_j^k, & \forall\, i,j,k\in[n], & j\geq i, & (56)\\
& & x_i^k + x_j^k - z_{ij}^k, \leq 1, & \forall\, i,j,k\in[n], & j\geq i, & (57)\\
& & z_{ij}^k = z_{ji}^k & \forall i,j,k\in[n], & j<i, & (58)\\
& & 0\leq x_i^k \leq 1 & \forall i,k\in[n], & & (59)\\
& & 0\leq z_{ij}^k \leq 1, & \forall\, i,j,k\in[n], & j\geq i. & (60)
\end{array}
$$

The disadvantage of this method is that it adds an order of n^3 constraints to the problem and n^3 new variables. While the high number of constraints and variables makes the formulation relatively strong, it also increases computation time. We can reduce the number of variables added by using a linearization method more adjusted to the QBPP.

3.1.2 Method of Fan and Pardalos

We let z_{ij} be a binary decision variable under the following constraints:

$$\begin{aligned}
z_{ij} + x_i^k - x_j^k &\leq 1, && \forall\, i,j,k \in [n], && j \geq i, && (61)\\
z_{ij} - x_i^k + x_j^k &\leq 1, && \forall\, i,j,k \in [n], && j \geq i, && (62)\\
-z_{ij} + x_i^k + x_j^k &\leq 1, && \forall\, i,j,k \in [n], && j \geq i, && (63)\\
z_{ij} &= z_{ji} && \forall\, i,j \in [n], && j < i, && (64)\\
0 \leq z_{ij} &\leq 1, && \forall\, i,j \in [n], && j \geq i. && (65)
\end{aligned}$$

Note that the variable z_{ij} is independent of the bin k. Thus if there is any bin that contains both item i and j, $z_{ij} = 1$. More specifically, if for some k, $x_i^k = 1$ and $x_j^k = 0$ then constraint (61) forces $z_{ij} = 0$. The same holds true for $x_i^k = 0$ and $x_j^k = 1$ with constraints (62). If both $x_i^k = 1$ and $x_j^k = 1$ then constraint (63) forces $z_{ij} = 1$. Suppose for some k that $x_i^k = 0$ and $x_j^k = 0$ then neither item is packed in bin k. This means that there must be some other bin which contains either item i or j or both. This means variable z_{ij} will be forced to take value 0 or 1 by the previously mentioned cases.

Because z_{ij} is independent of bin k we can rewrite the objective function to

$$\min \frac{1}{2} \sum_{i\in[n]} \sum_{j\in[n]} d_{ij} z_{ij} + \alpha \sum_{k\in[n]} y_k. \tag{66}$$

Thus we can add constraints (61) - (65) to F_{QP} with objective function (66) and get a new relaxation of the QBPP. We will call the method $F_{F\&P}$:

$$\begin{aligned}
F_{F\&P} \quad \min \quad & \frac{1}{2} \sum_{i\in[n]} \sum_{j\in[n]} d_{ij} z_{ij} + \alpha \sum_{k\in[n]} y_k && && && (67)\\
\text{s.t.} \quad & x_i^k \leq y_k && \forall i,k \in [n], && && (68)\\
& \sum_{k\in[n]} x_i^k = 1 && \forall i \in [n], && && (69)\\
& \sum_{i\in[n]} w_i x_i^k \leq W y_k && \forall k \in [n], && && (70)\\
& z_{ij} + x_i^k - x_j^k \leq 1, && \forall\, i,j,k \in [n], && j \geq i, && (71)\\
& z_{ij} - x_i^k + x_j^k \leq 1, && \forall\, i,j,k \in [n], && j \geq i, && (72)\\
& -z_{ij} + x_i^k + x_j^k \leq 1, && \forall\, i,j,k \in [n], && j \geq i, && (73)\\
& z_{ij} = z_{ji} && \forall\, i,j \in [n], && j < i, && (74)\\
& x_i^k \geq 0 && \forall i,k \in [n], && && (75)\\
& y_k \leq 1 && \forall k \in [n], && && (76)\\
& 0 \leq z_{ij} \leq 1, && \forall\, i,j \in [n], && j \geq i. && (77)
\end{aligned}$$

We can also then get the $F^{+}_{F\&P}$ method by applying symmetry reduction to the above formulation:

$$
\begin{aligned}
F^+_{F\&P} \quad & \min \quad \frac{1}{2}\sum_{i\in[n]}\sum_{j\in[n]} d_{ij}z_{ij} + \alpha \sum_{k\in[n]} x_k^k && && (78)\\
& \text{s.t.} \quad x_i^k \le x_k^k && \forall i,k \in [n], && (79)\\
& \sum_{k\in[n]} x_i^k = 1 && \forall i \in [n], && (80)\\
& \sum_{i\in[n]} w_i x_i^k \le W y_k && \forall k \in [n], && (81)\\
& x_i^k = 0 && \forall i,k \in [n], \quad k > i, && (82)\\
& z_{ij} + x_i^k - x_j^k \le 1, && \forall\, i,j,k \in [n], \quad j \ge i, && (83)\\
& z_{ij} - x_i^k + x_j^k \le 1, && \forall\, i,j,k \in [n], \quad j \ge i, && (84)\\
& -z_{ij} + x_i^k + x_j^k \le 1, && \forall\, i,j,k \in [n], \quad j \ge i, && (85)\\
& z_{ij} = z_{ji} && \forall\, i,j \in [n], \quad j < i, && (86)\\
& 0 \le x_i^k \le 1 && \forall i,k \in [n], && (87)\\
& 0 \le z_{ij} \le 1, && \forall\, i,j \in [n], \quad j \ge i. && (88)
\end{aligned}
$$

This method also adds an order of n^3 new constraints to the problem but does so using n^2 variables. Because the method has only one variable to indicate whether two items are in any bin together, as opposed to the n variables the method of Glover and Woolsey has, these formulations are weaker than their Glover and Woolsey counterparts. In the next method we will remove some of the added constraints.

3.1.3 Abridged Method of Fan and Pardalos

Constraints (62) force $z_{ij} = 0$ when there is a bin k with $x_i^k = 0$ and $x_j^k = 1$. However, because all items must be packed in a bin, there must now also be a bin $l : x_i^l = 1$ and $x_j^l = 0$. But if such a bin exists then constraints (61) force $z_{ij} = 0$. Thus we can remove constraints (62) while keeping the results for z_{ij} per case intact.

Thus by adding only constraints (61), (63) and (65) we still have a Linear relaxation of F_{QP} that can still give a good bound on the optimal value. Because we remove a constraint the new methods will be weaker than their regular Fan and Pardalos counterpart. We will call $F_{F\&P}$ without constraints (72) the $F_{AF\&P}$ formulation. We will use $F^+_{AF\&P}$ to refer to the $F^+_{F\&P}$ formulation without constraints (84). We state them below for completeness.

$$
\begin{aligned}
F_{AF\&P} \quad & \min \quad \frac{1}{2}\sum_{i\in[n]}\sum_{j\in[n]} d_{ij}z_{ij} + \alpha \sum_{k\in[n]} y_k && && (89)\\
& \text{s.t.} \quad x_i^k \le y_k && \forall i,k \in [n], && (90)\\
& \sum_{k\in[n]} x_i^k = 1 && \forall i \in [n], && (91)\\
& \sum_{i\in[n]} w_i x_i^k \le W y_k && \forall k \in [n], && (92)\\
& z_{ij} + x_i^k - x_j^k \le 1, && \forall\, i,j,k \in [n], \quad j \ge i, && (93)\\
& -z_{ij} + x_i^k + x_j^k \le 1, && \forall\, i,j,k \in [n], \quad j \ge i, && (94)\\
& z_{ij} = z_{ji} && \forall\, i,j \in [n], \quad j < i, && (95)\\
& x_i^k \ge 0 && \forall i,k \in [n], && (96)\\
& y_k \le 1 && \forall k \in [n], && (97)\\
& 0 \le z_{ij} \le 1, && \forall\, i,j \in [n], \quad j \ge i. && (98)
\end{aligned}
$$

$$
\begin{array}{llllll}
F^{+}_{AF\&P} & \min & \frac{1}{2}\sum_{i\in[n]}\sum_{j\in[n]} d_{ij}z_{ij} + \alpha \sum_{k\in[n]} x_k^k & & & (99) \\
 & \text{s.t.} & x_i^k \le x_k^k & \forall i,k\in[n], & & (100) \\
 & & \sum_{k\in[n]} x_i^k = 1 & \forall i\in[n], & & (101) \\
 & & \sum_{i\in[n]} w_i x_i^k \le W y_k & \forall k\in[n], & & (102) \\
 & & x_i^k = 0 & \forall i,k\in[n], & k > i, & (103) \\
 & & z_{ij} + x_i^k - x_j^k \le 1, & \forall\, i,j,k\in[n], & j \ge i, & (104) \\
 & & -z_{ij} + x_i^k + x_j^k \le 1, & \forall\, i,j,k\in[n], & j \ge i, & (105) \\
 & & z_{ij} = z_{ji} & \forall\, i,j\in[n], & j < i, & (106) \\
 & & 0 \le x_i^k \le 1 & \forall i,k\in[n], & & (107) \\
 & & 0 \le z_{ij} \le 1, & \forall\, i,j\in[n], & j \ge i. & (108)
\end{array}
$$

4 Semidefinite Problems

In this chapter we relax the QBPP to a positive semidefinite problem (SDP). We first give a short introduction of what a positive semidefinite matrix is and state some useful properties of positive semidefinite matrices. Finally we formulate the SDP relaxation of the QBBP.

4.1 Positive Semidefinite Matrices

Positive semidefinite matrices are commonly used in solving quadratic problems, such as in this article by Luo et al. [23] who use it to solve a unicast transit beamforming problem, or this article by Roupin [28] who applies SDP relaxations to the K-CLUSTER Problem, the Quadratic Assignment Problem and the Constrained-Memory Allocation Problem. A final example is this paper by de Meijer et al. [12] who applies semidefinite programming to relax the Quadratic Minimum Spanning Tree Problem. As they find SDP methods provide positive results, it would seem reasonable to apply SDP relaxations to the QBPP as well. To the best of our knowledge the method of using a SDP relaxation has not yet been applied to the QBPP, but because the structure of a positive semidefinite matrix lends itself naturally to quadratic problems and a positive semidefinite matrix has many useful properties we have reason to expect good results for this method. We list some of the properties of a semidefinite matrix below.

We first start with a definition of positive semidefinite matrices. A matrix $A \in \mathbb{S}^n$ is by definition positive semidefinite if

$$x^T A x \geq 0, \quad \forall\, x \in \mathbb{R}^n \backslash \{0\},$$

where $\mathbb{S}^n$ denotes the set of $n \times n$ symmetric real matrices. We denote a matrix being positive semidefinite by $A \succeq 0$, where $A \succeq B \Leftrightarrow A - B \succeq 0$.

Out of the above definition a couple useful properties arise:

- All eigenvalues of a positive semidefinite matrix are non-negative.
- The Cholesky decomposition [10]:

  $$\text{if and only if } A \succeq 0, \text{ then } \exists M \in \mathbb{R}^{n\times m} : A = MM^T.$$

  This means any positive semidefinite matrix must be the outer product of some vector with itself.
- If $A \succeq 0$ and a value on the diagonal of A is 0, then all values in that row and column of A must be 0.
- If $A \succeq 0$, then for any principal submatrix B, it holds that $B \succeq 0$.
- If $A \succeq 0$, then $\text{diag}(A) \geq 0$.
- A consequence of the Schur complement theorem (see e.g. [34]): if $b > 0$, C a matrix in $\mathbb{R}^{1\times m}$ and D a matrix in $\mathbb{R}^{m\times m}$ then $A = \begin{bmatrix} b & C \\ C^T & D \end{bmatrix}$ is positive semidefinite if and only if $D - C^T b^{-1} C \succeq 0$.

We see that the outer product of x^k with itself is a symmetric matrix that, due to the Cholesky decomposition, turns out to be positive semidefinite as well, thus the structure of our problem might lend itself well to semidefinite programming.

4.2 Introduction to Semidefinite Programming

First, SDPs use the trace inner product that is defined between two matrices:

$$C \bullet X = \sum_{i=1}^{n} \sum_{j=1}^{m} X_{ij} C_{ij}, \quad X, C \in \mathbb{R}^{n\times m}.$$

This can be seen as the sum over the element-wise multiplication of two matrices.

Next, we give the standard form of a semidefinite problem:

$$\begin{aligned} \min \quad & C \bullet X && && (109)\\ \text{s.t.} \quad & A_i \bullet X = b_i, && \text{for all}\, i \in \{1, ..., m\} && (110)\\ & X \succeq 0 && && (111) \end{aligned}$$

We can see that if we take a $X = \mathrm{Diag}(x)$, we reduce the problem to a LP with the vectors of the LP on the diagonal of the matrices of the SDP. Hence we can view the SDP as a generalisation of linear optimization problems.

For a thorough introduction into Semidefinite programming, the reader can refer to the book by Wolkowicz et al. [33]. For a more basic and recent text, we recommend the book by Skrzypczyk and Cavalcanti [32]. The book also discusses the many applications of semidefinite programming, such as in quantum physics or quantum information science. The modernity of these applications also shows why the field of semidefinite programming is of more recent interest, even though the field goes as far back as 1963 [2].

4.3 Relaxing the QBPP to a SDP

In this section we will reformulate the QBPP to a different program, that we can then relax to a SDP. Recall the QBPP in its F_{IQP} formulation, where x_i^k indicates whether item i is in bin k. The binary matrix $X^k = x_i^k (x_j^k)^T$ then denotes whether item i and j are together in bin k. Using this matrix, we can reformulate the QBPP as follows:

$$\begin{aligned} \min \quad & \frac{1}{2} \sum_{k \in [n]} (X^k \bullet C) + \alpha \sum_{k \in [n]} y_k && && (112)\\ \text{s.t.} \quad & x_i^k \leq y_k && \forall i, k \in [n], && (113)\\ & \sum_{k \in [n]} x_i^k = 1 && \forall i \in [n], && (114)\\ & \sum_{i \in [n]} w_i x_i^k \leq W y_k && \forall k \in [n], && (115)\\ & \mathrm{diag}(X^k) = x^k && \forall k \in [n], && (116)\\ & y_k \in \{0, 1\} && \forall k \in [n], && (117)\\ & X^k \succeq 0 && \forall k \in [n], && (118)\\ & X^k = x^k (x^k)^T && \forall k \in [n]. && (119) \end{aligned}$$

Most constraints are written the same as for the F_{IQP} for clarity's sake, but can trivially be rewritten using strictly the trace inner product. The objective function, previously containing $\frac{1}{2} \sum_{i \in [n]} \sum_{j \in [n]} \sum_{k \in [n]} d_{ij} x_i^k x_j^k$, is now more easily defined by the trace inner product. We have also replaced the binary constraints with constraints (116) and (118). We will show that constraints (116), (119) and (118) reduce to the binary constraints for x_i^k.

Because $x^k = \mathrm{diag}(X^k)$, we have that $x_i^k = X_{i,i}^k \quad \forall\, i, k \in [n]$. Then, because $X^k = x^k (x^k)^T$ we know that $X_{i,i}^k = (x_i^k)^2 \,\forall\, i, k \in [n]$. Combining these two statements we get

$$x_i^k = (x_i^k)^2 \to x_i^k = 1 \vee x_i^k = 0 \quad \forall\, i, k \in [n].$$

Which is equivalent to the binary constraints.

The above problem is not a SDP because of constraints (119) and (117), which cannot be rewritten in standard form. We can replace constraints (119) by the following:

$$\begin{aligned} X^k - x^k (x^k)^T &\succeq 0 && \forall\, k \in [n], && (120)\\ \mathrm{rank}(X^k) &= 1 && \forall\, k \in [n]. && (121) \end{aligned}$$

If constraints (120) and (118) hold, then by the Schur complement theorem this is equivalent to:

$$\begin{bmatrix} 1 & (x^k)^T \\ x^k & X^k \end{bmatrix} \succeq 0 \qquad \forall\, k \in [n]. \qquad (122)$$

Thus we can replace constraints (120) and (118) with constraints (122).

Finally, for the relaxation to a SDP we drop the non-convex constraints (121) and bound constraints (117) to end up with the following SDP:

$$\min \quad \frac{1}{2}\sum_{k\in[n]}(X^k \bullet C) + \alpha \sum_{k\in[n]} y_k \tag{123}$$

$$\text{s.t.} \quad x_i^k \le y_k \qquad \forall i,k \in [n], \tag{124}$$

$$\sum_{k\in[n]} x_i^k = 1 \qquad \forall i \in [n], \tag{125}$$

$$\sum_{i\in[n]} w_i x_i^k \le W y_k \qquad \forall k \in [n], \tag{126}$$

$$0 \le y_k \le 1 \qquad \forall k \in [n], \tag{127}$$

$$\operatorname{diag}(X^k) = x^k \qquad \forall k \in [n], \tag{128}$$

$$\begin{bmatrix} 1 & (x^k)^T \\ x^k & X^k \end{bmatrix} \succeq 0 \qquad \forall\, k \in [n]. \tag{129}$$

4.4 Strengthening the SDP Formulation

There are some constraints we can add to the above SDP formulation to make our formulation stronger. We list these below as theorems with proof.

Theorem 4.1 *The Constraints*

$$X^k \ge 0, \quad \forall\, k \in [n]$$

are valid for the program (112) - (119).

Proof $X^k \succeq 0 \to \operatorname{diag}(X^k) \ge 0$, then because $\operatorname{diag}(X^k) = x^k$, we find that $x^k \ge 0$. Then finally because $X^k = x^k(x^k)^T$ we see that X^k is the outer product of two nonnegative vectors, thus X^k must be nonnegative as well. $\square$

Theorem 4.2 *The Constraints*

$$X^k \le J, \quad \forall\, k \in [n]$$

are valid for the program (112) - (119).

Proof Because $X^k = x^k(x^k)^T$ and $x^k \le y^k \le 1$, we have that any element of $X^k \le 1$. Thus $X^k \le J$. $\square$

Theorem 4.3 *The Constraint*

$$n \le \left(\sum_{k=1}^{n} X^k\right) \bullet J \le n^2$$

is valid for the program (112) - (119).

Proof For the lower bound, $(\sum_{k=1}^n X^k) \bullet J = \sum_{k=1}^n \sum_{j=1}^n \sum_{i=1}^n X_{i,j}^k$. Because all elements of X^k are non-negative and we sum over all elements of the matrix we also sum over at least the diagonal of X^k. Thus we know that $\sum_{k=1}^n \sum_{j=1}^n \sum_{i=1}^n X_{i,j}^k \ge \sum_{i=1}^n \sum_{k=1}^n \operatorname{diag}(X^k)_i$ which, because of constraints (116) is equal to $\sum_{i=1}^n \sum_{k=1}^n x_i^k = \sum_{i=1}^n 1 = n$.

For the upper bound, first we rewrite $\sum_{k=1}^n \sum_{j=1}^n \sum_{i=1}^n X_{i,j}^k$ to $\sum_{k=1}^n \sum_{j=1}^n \sum_{i=1}^n x_i^k x_j^k$. Then because $x_j^k \le 1$, we have that

$$\sum_{k=1}^n \sum_{j=1}^n \sum_{i=1}^n x_i^k x_j^k \le \sum_{k=1}^n \sum_{j=1}^n \sum_{i=1}^n x_i^k = \sum_{k=1}^n \sum_{j=1}^n 1 = n^2. \ \square$$

Theorem 4.4 *The Constraints*

$$ww^T \bullet X^k \le W^2 y^k \quad \forall k \in [n]$$

are valid for the program (112) - (119).

Proof First we rewrite $ww^T \bullet X^k$ to $ww^T \bullet x^k(x^k)^T$ which is equal to $\sum_{j=1}^n \sum_{i=1}^n w_j w_i x_i^K x_j^k = \sum_{i=1}^n w_i x_i^k \sum_{j=1}^n w_j x_j^k$. Using constraints (115) we see that this is less than or equal to Wy^kWy^k. Then because y^k is a binary variable this is equal to W^2y^k. □

When adding constraints (4.1) - (4.4) we get the following program:

$$
\begin{array}{llrlr}
F_{SDP} & \min & \frac{1}{2}\sum_{k\in[n]}(X^k \bullet C) + \alpha \sum_{k\in[n]} y_k & & (130) \\
& \text{s.t.} & x_i^k \leq y_k & \forall i,k \in [n], & (131) \\
& & \sum_{k\in[n]} x_i^k = 1 & \forall i \in [n], & (132) \\
& & \sum_{i\in[n]} w_i x_i^k \leq W y_k & \forall k \in [n], & (133) \\
& & \operatorname{diag}(X^k) = x^k & \forall\, k \in [n], & (134) \\
& & \begin{bmatrix} 1 & (x^k)^T \\ x^k & X^k \end{bmatrix} \succeq 0 & \forall\, k \in [n], & (135) \\
& & 0 \leq y_k \leq 1 & \forall k \in [n], & (136) \\
& & X^k \geq 0 & \forall\, k \in [n], & (137) \\
& & X^x \leq J & \forall\, k \in [n], & (138) \\
& & n \leq (\sum_{k=1}^n X^k) \bullet J \leq n^2, & & (139) \\
& & ww^T \bullet X^k \leq W^2 y^k & \forall\, k \in [n]. & (140)
\end{array}
$$

We will call this formulation the F_{SDP} formulation.

Theorem 4.5 *The F_{SDP} formulation is stronger than the $F_{G\&W}$ formulation*

proof To prove the theorem we need to show that any feasible solution of F_{SDP} is also feasible for $F_{G\&W}$. Constraints (40) - (42) and (47) are equal so it trivial to show that the solution of F_{SDP} is feasible. Let X^k be a feasible solution of F_{SDP}. $0 \leq X^k \leq J$ and $\operatorname{diag}(X^k) = x^k$ thus $0 \leq x_i^k \leq 1 \forall\, i,k \in [n]$, which satisfies constraints (46).

Next we let $z_{ij}^k = x_i^k x_j^k$. Then because both x_i^k and x_j^k have values between 0 and 1, it is easy to see that constraints (43), (44) and (49) hold. Finally for constraints (45) we see that:

$$x_i^k + x_j^k - z_{ij}^k \leq 1 \Leftrightarrow x_i^k + x_j^k - x_i^k x_j^k \leq 1 \Leftrightarrow x_i^k(1 - x_j^k) \leq 1 - x_j^k.$$

If $x_j^k = 1$ then the statement is true. Otherwise we divide by $1 - x_j^k$ and find that $x_i^k \leq 1$ which we also know is true. □

4.4.1 Symmetry Reduction for the SDP Formulation

Similar to the LPs of Chapter 3 we can also apply symmetry reduction to the F_{SDP} formulation. This results in the F_{SDP}^+ formulation.

$$
\begin{aligned}
F^{+}_{SDP} \qquad \min \qquad & \frac{1}{2}\sum_{k\in[n]}(X^k \bullet C) + \alpha \sum_{k\in[n]} x^k_k && && (141)\\
\text{s.t.} \qquad & x^k_i \le x^k_k && \forall i,k \in [n], && (142)\\
& \sum_{k\in[n]} x^k_i = 1 && \forall i \in [n], && (143)\\
& \sum_{i\in[n]} w_i x^k_i \le W y_k && \forall k \in [n], && (144)\\
& x^k_i = 0 && \forall k \in [n],\, k > i, && (145)\\
& \operatorname{diag}(X^k) = x^k && \forall\, k \in [n], && (146)\\
& \begin{bmatrix} 1 & (x^k)^T \\ x^k & X^k \end{bmatrix} \succeq 0 && \forall\, k \in [n], && (147)\\
& X^k \ge 0 && \forall\, k \in [n], && (148)\\
& X^x \le J && \forall\, k \in [n], && (149)\\
& n \le (\sum_{k=1}^{n} X^k) \bullet J \le n^2, && && (150)\\
& ww^T \bullet X^k \le W^2 y^k && \forall\, k \in [n]. && (151)
\end{aligned}
$$

Similar to theorem 4.5 we can state that F^{+}_{SDP} is a stronger relaxation than $F^{+}_{G\&W}$.

5 Methods

We will compare 2 formulations and 10 different relaxations of the QBPP in terms of the tightness of their bounds and the time it takes to compute. The QBPP programs we use to compare with the relaxations are the F_{IQP} and F^{+}_{IQP} formulations. The relaxations we compare are the F_{QP} and F^{+}_{QP} formulations as in Chapter 2,the $F_{G\&W}$, $F_{F\&P}$, $F_{AF\&P}$, $F^{+}_{G\&W}$, $F^{+}_{F\&P}$ and $F^{+}_{AF\&P}$ as formulated in Chapter 3, and the F_{SDP} and F^{+}_{SDP} methods from Chapter 4.

We use a AMD Ryzen 5 3600 processor with 16GB of RAM. The programs were implemented in python using cvxpy 1.6 as the modelling package with a single core used. The SDP relaxations are solved using the Mosek 10.2.13 solver, the rest with the Gurobi 12.0.0 solver.

An instance of the QBPP is dependent on 5 variables, namely n, w, D, W and a, with D the dissimilarity matrix containing the values of d_{ij}. Because the dissimilarity values are what make the problem quadratic and more complex than the BPP, we will experiment with variations of D. We will vary D based on the sign of its values and sparsity. We will also slightly vary values of n to see how the different solvers compare when the size of the problem increases. To keep the number of experiments manageable in terms of time we will keep W and a the same across all experiments and use the same random distribution for w across all experiments.

For every experiment we set $W = 15$ and $a = 6$. The weights w will be uniformly randomly generated integers in [2,7]. This will result in an average weight of 4.5, which means on average roughly 3 items can be packed in a bin, resulting in enough space for the solution to pack multiple items in the same bin.

For our first experiment we will use $n = 15$. We will vary D based on sign in 3 scenarios.

- $D_{ij} > 0 \, \forall i, j \in [n]$, so items packed together will incur an extra cost. The values of D are uniformly randomly generated integers in $[1, 6]$.
- $D_{ij} < 0 \, \forall i, j \in [n]$, so items packed together will save costs, encouraging items to be packed together. The values of D are uniformly randomly generated integers in $[-6, -1]$.
- D is mixed, containing both positive and negative values. The values of D are uniformly randomly generated integers in $[-6, -1] \cup [1, 6]$.

We will also vary D based on sparsity across 4 different values, where D is either 0%, 25%, 50% or 75% sparse, which means a certain percentage of the values of D will be set to 0. We can view this as a measure of how quadratic an instance is, with a low sparsity making the problem more quadratic. We will run an extra case where D is 100% sparse, which means D is the 0-matrix, which reduces the problem to the Bin Packing Problem. This results in a total of 13 cases.

For each case we will run 10 instances, where for each instance we will get the optimal solution and computation time per program. Because the F_{IQP} and F^{+}_{IQP} method find a solution to the QBPP, as opposed to a solution to a relaxation, we can express the optimality gap as a percentage difference to the optimal solution, giving the relative optimality gap. The relative optimality gap is thus calculated as a percentage using $\frac{optimal-bound}{optimal} \cdot 100\%$ which means a low optimal value makes the percentages higher than with a higher optimal value, which can skew the results when we have different optimal values. For example $\frac{(200-100)}{100} \cdot 100\%$ and $\frac{2-1}{1} \cdot 100\%$ both indicate a gap of 100%, even though one method has an absolute gap of 100 and the other of 1. Because the optimal values can change significantly per method, we should be careful in drawing conclusions form the relative optimality gap in a vertical comparison. We can, however, compare these gaps perfectly fair horizontally, across the different formulations.

Because the relative optimality gap divides by the optimal solution of the QBPP, we need to make sure the optimal value is not 0. Thus we set $D_{ii} = 12 \forall \, i \in [n]$, to make sure our optimal solution has an unavoidable cost of $12n$. We did this for all cases except where D is 100% sparse.

The second experiment is the same as the first except with $n = 20$.

For the third and final experiment we will compare the best performing programs by how much we can increase n and still get an optimal result in a reasonable amount of time. We will compare them on the mixed dissimilarity set with 25% sparcity for a relatively average dataset. We will take 5 instances and again average the relative optimality gap percentage and computation time.

6 Results

In Tables 1 and 2 in the appendix we show the results for the first two experiments. Table 1 for $n = 15$ and Table 2 for $n = 20$. The columns indicate the method used. For each case we have two rows, where the top row indicates the relative optimality gap as a percentage and the bottom row indicates the computation time in seconds. The top case has a sparsity of 100%, and then we drop in sparsity per case, making the problem gradually more quadratic. The P in the left column indicates that D contains only positive values, the N indicates negative values and the M indicates mixed values as described in Chapter 5.

We see that across all methods the differences in sparsity and D have little effect on the result. The computation time across the different cases is very similar. Only for a D with negative values and low sparsity does the relative optimality gap increase significantly, especially for the F_{SDP} formulation (but still the lowest bound of all formulations). Because all the values of D are negative, packing as many different items in a bin is very advantageous, which is very easy if parts of an item can be packed like for all relaxations. This holds especially true for all LPs without symmetry constraint, who can pack potentially a part of every item uniformly into every bin. We should also note that the relative optimality gap may be higher because the optimal solution is lower due to the cost saving dissimilarities.

The F_{IQP} and F^+_{IQP} method perform very well. Of course they have a relative optimality gap of 0, as the solution they give is that of the QBPP. Surprisingly, the computation times compared to the other methods are very low, but both have some outliers where they take a very long time to compile, for example the (N, 25%) and (N, 0%) points for $n = 20$. This might be because the Gurobi solver cannot handle some cases of symmetry well, which the F^+_{IQP} method greatly reduces, which is why F^+_{IQP} performs more reliable than F_{IQP} in the extreme cases. Depending on the instance, symmetry reduction does not remove all duplicate optimal solutions, however, which could explain why the F^+_{IQP} method still has outliers. Thus even though the F^+_{IQP} method is structurally slightly slower than the F_{IQP} method, it is more reliable.

In terms of the relative optimality gap, we see that F_{QP} and F^+_{QP} perform the worst which is to be expected, as they add no additional constraints for the quadratic component. The biggest advantage of these methods is that they are very fast to compute, but their high relative optimality gap makes a branch-and-bound very expensive.

Compared to F_{QP}, the stronger linearized formulations $F_{G\&W}$, $F_{F\&P}$ and $F_{AF\&P}$ perform better as expected, surprisingly all giving the same relative optimality gap. Thus it appears the stronger formulations of $F_{G\&W}$ and $F_{F\&P}$ over $F_{AF\&P}$ have no effect on the optimal solution. These methods are not faster than their symmetry reduced counterparts and give a worse bound, making them inferior. The $F^+_{G\&W}$ does give a better bound than the $F^+_{F\&P}$ method as expected, which also gives a slightly better bound than the $F^+_{AF\&P}$ method. The $F^+_{AF\&P}$ is significantly faster than the other symmetry reduced LP methods, making it in our view the superior method.

The SDP formulations give a very tight bound for all instances, the F^+_{SDP} method in particular gives the tightest bound of all methods. The F^+_{SDP} program is only slightly slower while providing a significantly better bound than the F_{SDP} method. Unfortunately, the computation time is still higher than the F^+_{IQP} method, though both have no outliers, making them more reliable.

Overall we see that methods with symmetry reduction perform better than their counterparts. This could be because we make the solution space a lot smaller without eliminating an optimal solution, thus the computation time remains roughly the same. Because the solution space becomes smaller, however, we do get a tighter bound than without symmetry reduction.

We did the the first two experiments for different values of n and see very little pattern emerge, aside from all methods taking longer to compute. We can, however, use these experiments to estimate which methods can reasonably handle higher values of n.

The results for the third experiment can be found in Table 7. We increased n to higher values as is indicated in the first column. The F_{IQP} method started taking exponentially more time, while the F_{SDP} method increases in computation time a lot slower. We can also still see that the optimality gap for the F^+_{SDP} method remains tolerable, making this the superior method for higher values of n. Data had to be collected slowly for the final table, as it appeared the F^+_{IQP} method was slightly prone to taking too much time, probably similar to the high values we found in the first two experiments. For $n = 35$ the F_{IQP} method could not give an optimal solution anymore in a reasonable amount of time while the F_{SPD} worked fine.

Finally while the $F_{AF\&P}$ method works fast the gap does become very large, so it is unlikely it would be more useful in finding a solution for the QBPP than the F^+_{SDP} method.

7 Conclusion

In this thesis we described the QBPP and formulated different relaxations of the problem. We relaxed the QBPP to different LPs and SDPs and compared their strenghts mathematically. We then compared the computation time and bound of these methods in a simulation and found that the strengths we found indeed held up. Across all LP formulations we found that symmetry reduction had a positive impact on the optimality gap without significantly increasing computation time, which is as expected.

We explained why we believed the SDP relaxation showed promise and indeed we found the F_{IQP}^{+} method works good for smaller n while the F_{SDP}^{+} method works better for larger values of n. This is the main conclusion of this thesis.

There are some points that give room for further research. For example, our methods didn't include variations in W, a or w. This assured that there were always a certain number of items that could be packed together, but potentially with real data W could be relatively small, such that items rarely fit together, or W could be relatively big with a high a, strongly encouraging items to be packed together.

Because the SDP relaxations were strong we could have also included the F_{SDP} method in our comparison. It could be possible for different formulations of SDP relaxations give different bounds and computation times. An example would be removing the constraint on y^k, which was done in a simulation once by accident and gave a surprisingly strong bound.

We also didn't manage to increase n to levels that were to us satisfying. Unfortunately the Gurobi solver couldn't reliably solve the instances above a certain n, so further research could be done for bigger instances with the SDP relaxation. An obvious extension is then to also extend the SDP method with a brand-and-bound or similar method. as previously stated, Chagas [9]has done something similar with the linearisations, but because the F_{SDP}^{+} method currently turns out to be stronger than the linearisations we expect even stronger results with a branch-and-bound method.

A possible extension would be a problem where the capacity and cost of the problem are not uniform, adding complexity but more in line with real applications, where uniform cost and capacity of a bin is rare. Because the QBPP is relatively unknown research into heuristics could be interesting to develop a strong upper bound on an optimal solution.

Finally we hope that this thesis sparked interest among the readers for the QBPP. A simple extension to the BPP with surprising depth for its optimal solution.

References

[1] N. Aydın, İ. Muter, and Ş. Birbil. Multi-objective temporal bin packing problem: An application in cloud computing. *Computers & Operations Research*, 121, 2020.

[2] R. Bellman and K. Fan. On systems of linear inequalities in hermitian matrix variables. In *vol. 7 of Proceedings of Symposia in Pure Mathematics*, 1963.

[3] D. Bergman. An exact algorithm for the quadratic multiknapsack problem with an application to event seating. *INFORMS Journal on Computing*, 31(3):477 – 492, 2019.

[4] P. Bonami, V. Nguyen, M. Klein, and M. Minoux. On the solution of a graph partitioning problem under capacity constraints. In *Combinatorial Optimization*, 2012.

[5] D. Brown. A lower bound for on-line one-dimensional bin packing algorithms. Technical report, Coordinated Science Laboratory University of Illinois, 1979.

[6] M. Brusco and S. Stahl. *Branch-and-Bound Applications in Combinatorial Data Analysis.* Springer New York, NY, 2005.

[7] V. Cacchiani, M. Iori, A. Locatelli, and S. Martello. Knapsack problems — an overview of recent advances. part II: Multiple, multidimensional, and quadratic knapsack problems. *Computers & Operations Research*, 143, 2022.

[8] A. Caprara. Constrained 0–1 quadratic programming: Basic approaches and extensions. *European Journal of Operational Research*, 187(3):1494–1503, 2008.

[9] V. Chagas. Exact algorithms for the quadratic bin packing problem. Master's thesis, Universidade Estadual de Campinas, 2021.

[10] A. Cholesky. Note sur une méthode de résolution des équations normales provenant de l'application de la méthode des moindres carrés a un système d'équations linéaires en nombre inférieur a celui des inconnues. — application de la méthode a la résolution d'un système defini d'équations linéaires. *Bulletin géodésique*, 2:66–77, 1924.

[11] H. Christensen, A. Khan, S. Pokutta, and P. Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017.

[12] F. de Meijer, M. Siebenhofer, R. Sotirov, and A. Wiegele. Spanning and splitting: Integer semidefinite programming for the quadratic minimum spanning tree problem, 2024.

[13] M. Delorme, M. Iori, and S. Martello. Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research*, 255(1):1–20, 2016.

[14] N. Fan and P. Pardalos. Linear and quadratic programming approaches for the general graph partitioning problem. *Journal of Global Optimization*, 48:57 – 71, 2010.

[15] G. Gallo, P. Hammer, and B. Simeone. *Combinatorial Optimization*, chapter Quadratic knapsack problems, pages 132–149. Springer Berlin Heidelberg, Berlin, Heidelberg, 1980.

[16] M. R. Garey and D. S. Johnson. *Computers and intractability.* W.H.Freeman & Co Ltd, 1979.

[17] F. Glover and E. Woolsey. Converting the 0-1 polynomial programming problem to a 0-1 linear program. *Operations Research*, 22(1):180–182, 1974.

[18] B. Han, G. Diehr, and J. Cook. Multiple-type, two-dimensional bin packing problems: Applications and algorithms. *Annals of Operations Research*, 50:239–261, 1994.

[19] A. Hiley and B. Julstrom. The quadratic multiple knapsack problem and three heuristic approaches to it. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO '06, page 547–552, New York, NY, USA, 2006. Association for Computing Machinery.

[20] L. Huang, X. Chen, W. Huo, J. Wang, F. Zhang, B. Bai, and L. Shi. Branch and bound in mixed integer linear programming problems: A survey of techniques and trends. *CoRR*, 2021.

[21] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and Algorithms*, pages 449–465. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[22] A. Laurent and N. Klement. Bin packing problem with priorities and incompatibilities using pso: application in a health care community. *IFAC-PapersOnLine*, 52(13):2596–2601, 2019.

[23] Z. Luo, T. Chang, D. Palomar, and Y. Eldar. *Convex Optimization in Signal Processing and Communications*, chapter SDP relaxation of homogeneous quadratic optimization: approximation bounds and applications, pages 117 – 165. Cambridge Univ. Press Cambridge, UK, 2010.

[24] F. Margot. *50 Years of Integer Programming 1958-2008*, chapter Symmetry in Integer Linear Programming. Springer, Berlin, Heidelberg, 2010.

[25] C. Munien and A. Ezugwu. Metaheuristic algorithms for one-dimensional bin-packing problems: A survey of recent advances and applications. *Journal of Intelligent Systems*, 30(1):636–663, 2021.

[26] P. Pardalos, Y. Ye, and C. Han. Algorithms for the solution of quadratic knapsack problems. *Linear Algebra and its Applications*, 152:69–91, 1991.

[27] D. Pisinger. The quadratic knapsack problem—a survey. *Discrete Applied Mathematics*, 155(5):623–648, 2007.

[28] F. Roupin. From linear to semidefinite programming: An algorithm to obtain semidefinite relaxations for bivalent quadratic problems. *Journal of Combinatorial Optimization*, 8:469–493, 2004.

[29] R. Sadykov and F. Vanderbeck. Bin packing with conflicts: A generic branch-and-price algorithm. *INFORMS Journal on Computing*, 25(2):244 – 255, 2012.

[30] T. Saraç and A. Sipahioglu. Generalized quadratic multiple knapsack problem and two solution approaches. *Computers & Operations Research*, 43:78–89, 2014.

[31] G. Scheithauer. *Introduction to Cutting and Packing Optimization: Problems, Modeling Approaches, Solution Methods.* Springer International Publishing, 2018.

[32] P. Skrzypczyk and D. Cavalcanti. *Semidefinite Programming in Quantum Information Science.* IOP Publishing, 2023.

[33] H. Wolkowicz, R. Saigal, and L. Vandenberghe. *Handbook of Semidefinite Programming.* Springer New York, NY, 2000.

[34] F. Zhang. *The Schur complement and its applications.* Springer New York, NY, 2006.

Appendix

Table 1: Results for n=15

	F_{IQP}	F^{+}_{IQP}	F_{QP}	F^{+}_{QP}	$F_{G\&W}$	$F^{+}_{G\&W}$	$F_{F\&P}$	$F^{+}_{F\&P}$	$F_{AF\&P}$	$F^{+}_{AF\&P}$	F_{SDP}	F^{+}_{SDP}
(100%)	0	0	30	44	8	8	8	8	8	8	8	8
	0.1	0.21	0.12	0.16	7.04	6.97	3.77	3.59	2.46	2.5	3.69	3.92
(P, 75%)	0	0	165	137	87	81	87	87	87	87	1	1
	0.11	0.25	0.09	0.19	7.01	6.98	3.68	3.76	2.55	2.43	3.75	3.85
(P, 50%)	0	0	215	177	88	82	88	88	88	88	1	1
	0.16	0.19	0.1	0.16	6.95	6.91	3.78	3.64	2.44	2.48	3.72	3.78
(P, 25%)	0	0	243	199	88	82	88	88	88	88	2	2
	0.25	0.2	0.09	0.19	7.02	6.94	3.8	3.59	2.51	2.46	3.77	3.74
(P, 0%)	0	0	276	223	88	81	88	87	88	87	6	5
	0.85	0.22	0.1	0.16	7.02	6.96	3.7	3.8	2.46	2.51	3.77	3.83
(N, 75%)	0	0	165	137	87	81	87	87	87	87	1	1
	0.11	0.2	0.1	0.18	6.77	6.79	3.61	3.63	2.41	2.43	3.57	3.79
(N, 50%)	0	0	234	192	138	108	138	131	138	132	29	17
	0.52	0.22	0.1	0.18	6.88	6.68	3.62	3.65	2.45	2.36	3.62	3.87
(N, 25%)	0	0	272	225	208	146	208	190	208	192	84	49
	1.95	0.36	0.09	0.18	6.89	6.73	3.62	3.66	2.48	2.37	3.63	3.93
(N, 0%)	0	0	285	234	272	178	272	240	272	244	143	85
	12.18	0.43	0.1	0.19	7.05	6.86	3.71	3.73	2.49	2.4	3.66	3.9
(M, 75%)	0	0	165	137	87	81	87	87	87	87	1	1
	0.11	0.21	0.09	0.18	7.06	6.94	3.8	3.61	2.5	2.48	3.71	3.79
(M, 50%)	0	0	221	179	110	95	110	107	110	108	9	6
	0.4	0.19	0.09	0.19	7	6.97	3.57	3.74	2.5	2.51	3.71	3.8
(M, 25%)	0	0	252	206	139	109	139	132	139	132	15	11
	0.64	0.22	0.09	0.16	6.9	7.25	3.52	3.47	2.43	2.45	3.61	3.83
(M, 0%)	0	0	286	233	173	127	173	160	173	162	22	15
	1.22	0.31	0.09	0.19	6.97	6.9	3.75	3.78	2.49	2.6	3.7	3.86

Table 2: Results for n=20

	F_{IQP}	F^+_{IQP}	F_{QP}	F^+_{QP}	$F_{G\&W}$	$F^+_{G\&W}$	$F_{F\&P}$	$F^+_{F\&P}$	$F_{AF\&P}$	$F^+_{AF\&P}$	F_{SDP}	F^+_{SDP}
(100%)	0	0	30	46	5	5	5	5	5	5	5	5
	0.28	0.76	0.17	0.3	22.29	22.3	9.28	9.51	6.15	6.12	10.55	11.37
(P, 75%)	0	0	168	143	87	83	87	87	87	87	1	1
	0.18	0.51	0.16	0.27	22.86	22.81	9.63	10.01	6.24	6.22	10.55	11.38
(P, 50%)	0	0	226	189	87	82	87	87	87	87	1	1
	0.29	0.45	0.16	0.28	22.47	22.51	9.48	9.76	6.14	6.11	10.69	11.45
(P, 25%)	0	0	277	231	87	83	87	87	87	87	1	1
	0.61	0.62	0.17	0.28	22.72	22.89	9.51	9.77	6.19	6.1	10.77	11.38
(P, 0%)	0	0	325	269	87	83	87	87	87	87	4	4
	2.75	0.4	0.17	0.28	23.72	23.75	9.78	10.11	6.36	6.3	10.82	11.55
(N, 75%)	0	0	167	142	87	83	87	87	87	87	1	1
	0.22	0.75	0.18	0.3	23.19	22.77	9.44	9.61	6.17	6.28	10.54	11.25
(N, 50%)	0	0	264	221	166	122	166	155	166	157	51	28
	3.74	0.7	0.19	0.31	23.17	22.81	9.47	9.65	6.18	6.29	10.55	11.41
(N, 25%)	0	0	291	244	252	165	252	226	252	230	126	73
	95.02	2.73	0.21	0.32	23.33	23.43	9.94	10	6.31	6.4	10.65	11.77
(N, 0%)	0	0	329	279	369	239	369	325	369	332	224	137
	1225.25	10.52	0.21	0.33	24.63	24.59	10.23	10.38	6.42	6.6	10.73	12.07
(M, 75%)	0	0	167	141	87	82	87	87	87	87	1	1
	0.22	0.58	0.16	0.28	22.73	22.73	9.53	9.88	6.2	6.15	10.57	11.48
(M, 50%)	0	0	239	200	115	100	115	113	115	114	10	8
	0.87	0.48	0.16	0.28	22.62	22.7	9.48	9.8	6.13	6.11	10.85	11.43
(M, 25%)	0	0	275	228	153	115	153	143	153	145	20	13
	3.88	0.57	0.17	0.28	23.01	23.1	9.59	9.95	6.24	6.23	10.92	11.59
(M, 0%)	0	0	314	261	199	138	199	182	199	184	26	17
	38.76	0.84	0.17	0.28	23.95	23.97	9.93	10.3	6.38	6.36	10.96	11.5

Table 3: Results for higher values of n

n	F^+_{IQP} gap	F^+_{IQP} time	$F^+_{AF\&P}$gap	$F^+_{AF\&P}$ time	F^+_{SDP} gap	F^+_{SDP} time
20	0	0.51	175.7	6.18	20.8	11.3
21	0	1.08	191.2	7.67	21.7	14.1
22	0	1.42	192.4	9.21	20.2	16.9
23	0	1.00	205.3	9.67	24.3	20.1
24	0	2.30	212.3	12.2	20.6	23.2
25	0	7.88	221.8	14.1	23.3	27.7
26	0	7.55	228.3	15.4	26.6	32.8
27	0	4.35	211.9	18.5	25.1	38.2
28	0	7.25	233.7	21.0	26.1	42.9
29	0	16.8	230.3	23.3	28.6	49.4
30	0	13.2	236.3	27.1	28.3	57.4
31	0	33.0	252.4	29.8	28.9	67.1
32	0	59.9	268.9	35.4	32.7	76.7
33	0	74.7	273.7	40.9	37.5	85.3
34	0	237.2	280.6	49.0	35.9	97.4