A Novel SDP Relaxation for the Quadratic Assignment Problem using Cut Pseudo Bases

Maximilian John and Andreas Karrenbauer

Max Planck Institute for Informatics, Saarbrücken, Germany,
`firstname.lastname@mpi-inf.mpg.de`

Abstract The quadratic assignment problem (QAP) is one of the hardest combinatorial optimization problems. Its range of applications is wide, including facility location, keyboard layout, and various other domains. The key success factor of specialized branch-and-bound frameworks for minimizing QAPs is an efficient implementation of a strong lower bound. In this paper, we propose a lower-bound-preserving transformation of a QAP to a different quadratic problem that allows for small and efficiently solvable SDP relaxations. This transformation is self-tightening in a branch-and-bound process.

Keywords: quadratic assignment, semidefinite program, lower bound, branch and bound

1 Introduction

Assignment problems are some of the best-studied problems in combinatorial optimization, the task being to find a one-to-one correspondence of n items and n locations, i.e., an

$$x \in \mathit{\Pi}_n := \left\{ X \in \mathbb{Z}^{n\times n} : \sum_{i=1}^{n} x_{ik} = 1\, \forall k \in [n] \text{ and } \sum_{k=1}^{n} x_{ik} = 1\, \forall i \in [n] \right\},$$

such that the total cost of the assignment is minimized. If the objective function is linear, i.e., of the form $\sum_{i,k} c_{ik} x_{ik}$, the optimum can be computed efficiently due to Birkhoff's theorem [1], e.g., with the Hungarian method [2] in $\mathcal{O}(n^3)$, or with the bipartite matching algorithm of Duan and Su [3] for integer costs of at most C in $O(n^{5/2} \log C)$. However, the linear objective function restricts the modeling power because it does not account for the interaction between the items nor for the interactions between the locations. Therefore, Koopmans and Beckmann investigated a variant of a quadratic objective function of the form $\sum_{i,j,k,\ell} c_{ijk\ell} x_{ik} x_{j\ell}$ that also considers pair-wise dependencies of the input objects [4]. In their variant, the cost factors into dependencies between items and dependencies between locations, respectively. That is, $c_{ijk\ell} = f_{ij} \cdot d_{k\ell}$, or $C = F \otimes D$ in matrix notation using the Kronecker product. This variant of the QAP offers various practical applications such as the facility location problem [5] or the keyboard layout problem [6]. Moreover, it generalizes the traveling salesman problem [7] and several further real-life combinatorial problems such as the wiring problem [8] or hospital layout [9,10]. However, the QAP is very hard to solve even for small instances ($n \geq 30$). For example, the problem library QAPLIB [11] still contains decades-old unsolved instances and ones that were solved only recently by newly proposed techniques and/or the usage of massive computational power [12]. In the time of writing, its smallest unsolved instance consists of just 30 items and locations. On the theoretical side, Queyranne showed that the QAP is NP-hard to approximate within any constant factor, even if the cost can be factorized to a symmetric block diagonal flow matrix and a distance matrix describing the distances of a set of points on a line [13]. Detailed surveys on the Quadratic Assignment Problem can be found in [14,15,16].

A systematic approach for solving a QAP is to compute relaxations in a branch-and-bound framework. One of the earliest published lower bounds, the Gilmore-Lawler bound [17,18] for the Koopmans-Beckmann variant, reduces the problem to a linear assignment problem. However, it detoriates quickly with increasing instance sizes [19]. On the other hand, already the first level of the reformulation linearization techinque (RLT) by Frieze and Yadegar [20] produces strong lower bounds. But this comes at the expense of introducing n^4 many binary variables $y_{ijk\ell}$, i.e., one for each quadratic term occurring in the objective function. Thus, it takes a lot of resources (both in terms of CPU and RAM) to solve LP-relaxations for instances of practical input size. In contrast, the formulation of Kaufman and Broeckx [21] only contains $\mathcal{O}(n^2)$ variables, and thus, its LP-relaxation can be solved efficiently. Moreover, the primal heuristics of state-of-the-art MIP-solvers are able to quickly produce strong incumbents with this formulation. However, the lower bounds obtained by relaxing the integrality constraints of this formulation are very weak such that they often do not even surpass the trivial lower bound of $\sum_{i,j} \min\{c_{ijk\ell} : k, \ell \in [n]\}$ in reasonable time, which makes it impractical to use this formulation alone to close the gap between upper and lower bounds in a branch-and-bound process.

Furthermore, there are various relaxations of the QAP as a semidefinite program (SDP). For example, SDP-relaxations for the non-convex constraint $Y = X \otimes X$ were introduced in [22,23]. Recent approaches (e.g., [24]) have shown that these approaches can often efficiently produce good lower bounds for the QAP and beat common linear relaxations. We follow a different approach since we do not derive our SDP from this formulation, but transform the QAP to a different quadratic problem beforehand. In that sense, our approach is somewhat orthogonal to recent other SDP relaxations.

1.1 Our contribution

In this paper, we propose a novel SDP derived from a lower-bound-preserving transformation of a QAP instance to an auxiliary quadratic minimization problem with only $\mathcal{O}(n \log n)$ variables. SDP-relaxations with that few variables can be solved efficiently with modern interior point methods for conic optimization problems. Moreover, it is straight forward to integrate our relaxation in a branch-and-bound framework. While branching on single assignment variables typically results in very unbalanced branch-and-bound trees, our approach avoids this by design. To this end, we introduce the concept of cut pseudo bases, which has not been used — to the best of our knowledge — in this context before. Our goal was to develop an approach that still works with limited computational resources, e.g., on a laptop, for the cases when the lower bounds provided by Kaufman-Broeckx are too weak and when it is already infeasible to solve the LP-relaxation of RLT1. Furthermore, we present experimental results for instances with $n \geq 25$ in which we outperform both lower bounds mentioned above in terms of efficiency and effectiveness. The bounds produced by our SDP always exceed — just by construction — the trivial lower bound mentioned above.

2 A novel lower bound using SDP

Let n denote, throughout this paper, the respective number of items and locations. We assume for the sake of presentation that n is a power of 2. This is not a restriction because we can pad n with dummy items and locations. Moreover, the dummy items can be projected out easily in an implementation so that this also does not harm its performance.

The derivation is done in two steps. First, we design a new quadratic program that lower bounds the QAP and allows for a balanced branching tree. In the second step, we relax the new problem to an SDP.

Concerning the goal of achieving balanced branching trees, we revisit the well-known problem of branching on single assignment variables. Setting x_{ik} to 1 means fixing item i to location j, which is a very strong decision that affects all other variables in the i-th row or k-th column, forcing them to 0. On the other hand, if we set x_{ik} to 0, we just decide not to fix i to j. However, there are still $n-1$ other possible locations for i, so we basically did not decide much. This yields highly imbalanced branching trees as it is much more likely to prune in the 1-branches of a branch-and-bound process. This undesirable effect can be avoided by the well-known idea of generalized upper-bound branching (see Section 7 of [25]). Inspired by this, we consider a similar approach illustrated in the following IP formulation with n auxiliary z-variables:

$$\begin{array}{lllr}
\text{minimize} & \sum\limits_{i,j,k,\ell=1}^{n} c_{ijk\ell} x_{ik} x_{jl} & & \\
\text{s.t.} & \sum\limits_{i=1}^{n} x_{ik} = 1 & & \forall k \in [n] \\
& \sum\limits_{k=1}^{n/2} x_{ik} = z_i & \sum\limits_{k=n/2+1}^{n} x_{ik} = 1 - z_i & \forall i \in [n] \\
& x_{ik} \in \{0,1\} & & \forall i,k \in [n] \\
& z_i \in \{0,1\} & & \forall i \in [n].
\end{array}$$

If we branch on the z-variables instead of the assignment variables, our branching tree is much more likely to be balanced because either choice is equally strong. However, it is not sufficient to branch only on these z-variables because too many degrees of freedom still remain open even after all z-variables are set.

If we want to completely determine the x-variables and thus be able to project them out, we should introduce further binary z-variables. To this end, we introduce *cut pseudo bases*.

2.1 Introduction of cut pseudo bases

The key idea of cut pseudo bases is the usage of cuts in the complete graph with the locations as nodes. Consider a balanced subset of the nodes, i.e., one of size $n/2$. Instead of assigning an item to a certain location, we now assign it to one of the "halves" of the location space. We repeat this cutting of the location space until we reach a state where — after a finite number of assignments — every item can be uniquely mapped to a single location. Moreover, we cut the space in a balanced way, i.e., we require that each side of the cut is equally large. Let us formalize these requirements.

Definition 1. *A set of cuts over the location space such that*

- *all cuts are balanced,*
- *all singleton locations can be expressed by a linear combination of cuts, and*
- *it is inclusion-wise minimal*

is called a cut pseudo base*.*

Clearly, the size of a cut pseudo base is $\log_2 n$ when n is a power of 2 and thus $\lceil \log_2 n \rceil$ in general by the padding argument. To illustrate the concept of a pseudo base, consider the following example.

Example 1. Enumerate the $n = 2^k$ locations by $0, \ldots, n-1$, and consider the binary decomposition of these numbers. For every bit $b = 0, \ldots, k-1$, we define a cut that separates all locations with numbers differing in the b-th bit. Then, this collection of cuts forms a cut pseudo base.

Note that any arbitrary cut pseudo base can be transformed to the binary decomposition pseudo base by a permutation of the locations. Hence, we will employ this cut pseudo base as a reference throughout this paper for the sake of presentation and simplicity.

2.2 Exchanging assignment variables by cut variables

The cut pseudo bases introduced in the previous subsection are balanced by definition, meaning that assigning an item to one side of a cut in the pseudo base is just as effective as assigning it to the other side. However, the decision of whether a particular item should be assigned to a fixed location is highly unbalanced, as we have already discussed. Hence, our goal is to get rid of the assignment variables and introduce the cut variables instead. This will also benefit the number of binary variables which decreases from n^2 assignment variables to $n \log_2 n$ cut variables. Let $(S_b)_{b \in [B]}$ be an arbitrary but fixed cut pseudo base. Note that $B = \lceil \log_2 n \rceil$, otherwise, (S_b) cannot be a cut pseudo base. We introduce the variables $z_i^b \in \{0,1\}$ for every cut S_b, indicating whether i is assigned to the 0- or 1-side of the cut S_b, i.e., to the outside or the inside, respectively. We relate them to the assignment variables in the following manner.

$$x_{ij} = 1 \Leftrightarrow \forall b \in [B] \;\; j \in z_i^b\text{-side of cut } S_b$$

We consider an arbitrary cut b in the following and omit the superscript b to simplify the notation and thereby improve readability. Observe that

$$z_i z_j + z_i(1-z_j) + (1-z_i)z_j + (1-z_i)(1-z_j) = 1$$

holds for any $z_i, z_j \in \mathbb{R}$ and that for a binary solution exactly one of the four terms is 1, and the others vanish.

Thus, we obtain for any assignment x and the corresponding binary z-variables that

$$\begin{aligned}
&\textstyle\sum_{\substack{i,j,\\k,\ell}} c_{ijk\ell} x_{ik} x_{j\ell} \\
&\quad = \textstyle\sum_{i,j} [z_i z_j + z_i(1-z_j) + (1-z_i)z_j + (1-z_i)(1-z_j)] \sum_{k,\ell} c_{ijk\ell} x_{ik} x_{j\ell} \\
&\quad \geq \textstyle\sum_{i,j} z_i z_j \cdot \min\{\sum_{k,\ell} c_{ijk\ell} x_{ik} x_{j\ell} : x \in \Pi_{ij}^{(11)}\} \\
&\quad + \textstyle\sum_{i,j} z_i (1-z_j) \cdot \min\{\sum_{k,\ell} c_{ijk\ell} x_{ik} x_{j\ell} : x \in \Pi_{ij}^{(10)}\} \\
&\quad + \textstyle\sum_{i,j} (1-z_i) z_j \cdot \min\{\sum_{k,\ell} c_{ijk\ell} x_{ik} x_{j\ell} : x \in \Pi_{ij}^{(01)}\} \\
&\quad + \textstyle\sum_{i,j} (1-z_i)(1-z_j) \cdot \min\{\sum_{k,\ell} c_{ijk\ell} x_{ik} x_{j\ell} : x \in \Pi_{ij}^{(00)}\}
\end{aligned}$$

where $\Pi_{ij}^{(11)}$ denotes the set of all assignments in which i and j are both assigned inside the cut, where $\Pi_{ij}^{(10)}$ denotes the set of all assignments in which i is assigned inside the cut and j is assigned to the outside, and so on. In the following, we argue that this is indeed a valid lower bound. To this end, let $c_{ij}^{(\alpha\beta)} := \min\{\sum_{k,\ell} c_{ijk\ell} x_{ik} x_{j\ell} : x \in \Pi_{ij}^{(\alpha\beta)}\}$ denote the optimum objective values of the corresponding optimization problems for $\alpha, \beta \in \{0,1\}$, and observe that $c_{ij}^{(\alpha\beta)}$ only contributes to the right-hand side if $z_i = \alpha$ and $z_j = \beta$.

This yields an objective function that is free of x-variables. Furthermore, the minimum of the original objective taken over all $x \in \Pi$ is bounded from below by the minimum over all z that determine an assignment.

At first glance, it seems that we have to solve $4n^2$ QAPs to compute the coefficients for the new objective function. However, a close inspection of the subproblems reveals that $c_{ij}^{(\alpha\beta)}$ is determined by the minimum $c_{ijk\ell}$ over all k, ℓ such that the b-th bits of k and ℓ are α and β, respectively. This can be computed efficiently for each pair ij by a single scan over all $c_{ijk\ell}$. Note that in the Koopmans-Beckmann variant of a QAP, we have $c_{ijk\ell} = f_{ij} \cdot d_{k\ell}$, and thus, it suffices to scan over the distance pairs $d_{k\ell}$ of the locations k and ℓ. Furthermore, such a single scan can also take additional constraints into account, e.g., excluded pairs due to a branching process. Hence, the lower bound of our approach is self-tightening in a branch-and-bound process. In every branching step, we can update our cost estimation for this particular setting of excluded pairs.

2.3 Towards an SDP

In order to obtain a reasonable SDP relaxation, we apply the typical transformation to map $\{0,1\}$-variables to $\{-1,1\}$-variables. That is, we use the linear transformation $z_i = \frac{1+y_i}{2}$. This implies that $1 - z_i = \frac{1-y_i}{2}$. Plugging this into

$$c_{ij}^{(11)} z_i z_j + c_{ij}^{(10)} z_i (1-z_j) + c_{ij}^{(01)} (1-z_i) z_j + c_{ij}^{(00)} (1-z_i)(1-z_j)$$

yields

$$\begin{aligned}\sum_{\alpha,\beta=0}^{1} c_{ij}^{(\alpha\beta)} \cdot \tfrac{1-(-1)^{\alpha} y_i}{2} \cdot \tfrac{1-(-1)^{\beta} y_j}{2} &= \sum_{\alpha,\beta=0}^{1} c_{ij}^{(\alpha\beta)} \cdot \tfrac{1-(-1)^{\alpha} y_i - (-1)^{\beta} y_j + (-1)^{\alpha+\beta} y_i y_j}{4} \\ &= \tfrac{c_{ij}^{(11)}+c_{ij}^{(10)}+c_{ij}^{(01)}+c_{ij}^{(00)}}{4} + \tfrac{c_{ij}^{(11)}+c_{ij}^{(10)}-c_{ij}^{(01)}-c_{ij}^{(00)}}{4} \cdot y_i \\ &+ \tfrac{c_{ij}^{(11)}-c_{ij}^{(10)}+c_{ij}^{(01)}-c_{ij}^{(00)}}{4} \cdot y_j + \tfrac{c_{ij}^{(11)}-c_{ij}^{(10)}-c_{ij}^{(01)}+c_{ij}^{(00)}}{4} \cdot y_i y_j .\end{aligned}$$

We separate and symmetrize the constant, linear, and quadratic terms such that we can write the total sum over all i, j in matrix-vector notation as

$$y^T C y + c^T y + \gamma$$

with

$$\begin{aligned} C_{ij} &:= \frac{c_{ij}^{(11)} + c_{ji}^{(11)} - c_{ij}^{(10)} - c_{ji}^{(10)} - c_{ij}^{(01)} - c_{ji}^{(01)} + c_{ij}^{(00)} + c_{ji}^{(00)}}{8} \\ c_i &:= \frac{1}{4} \sum_{j=1}^{n} c_{ij}^{(11)} + c_{ji}^{(11)} + c_{ij}^{(10)} - c_{ji}^{(10)} - c_{ij}^{(01)} + c_{ji}^{(01)} - c_{ij}^{(00)} - c_{ji}^{(00)} \\ \gamma &:= \frac{1}{4} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^{(11)} + c_{ij}^{(10)} + c_{ij}^{(01)} + c_{ij}^{(00)} . \end{aligned}$$

To relax the quadratic part in the objective using a semidefinite matrix, we use a standard fact about the trace, i.e., $y^T C y = \mathrm{tr}(y^T C y) = \mathrm{tr}(C y y^T)$. Thus, we replace the quadratic term $y^T C y$ in the objective function by the Frobenius product[1] $C \bullet Y$ and hope that $Y = yy^T$. However, such a rank-1-constraint is not convex, and we relax it to $Y \succcurlyeq yy^T$, which means that $Y - yy^T$ is positive semi-definite. Since the latter is a Schur complement, this condition is equivalent to

$$\begin{pmatrix} 1 & y^T \\ y & Y \end{pmatrix} \succcurlyeq 0.$$

To accomplish this, we could augment the matrix Y by a 0-th row and column, or we could also use an item that has already been fixed w.r.t. the side of the cut b under consideration, e.g., use one of the dummy items introduced to fill

[1] The Frobenius product $A \bullet B := \mathrm{tr}(A^T B) = \sum_{i,j} a_{ij} b_{ij}$ is the standard inner product on the space of $n \times n$ matrices used in semi-definite programming.

up the number of items to a power of 2. That is, if $y_i = 1$ is already fix for some item i, we may require $Y_{ij} = y_j$ for all j and $Y \succcurlyeq 0$. The former constraint can be written as $e_i e_j^T \bullet Y - e_j^T y = 0$, which modern SDP solvers such as Mosek [26] directly allow without a transformation to an equivalent block-diagonal pure SDP formulation. If $y_i = -1$, we obtain the constraints $e_i e_j^T \bullet Y + e_j^T y = 0$ for all items j instead.

In the following, we list further constraints that we may add to the SDP to improve the strength of the lower bound on the QAP. Recall that we omitted any superscripts to identify the cut under consideration. However, we will argue with the complete cut pseudo base in the following, so we use Y^b for the matrix corresponding to cut b and y^b to identify the linear terms corresponding to this cut. Similarly, we shall use C^b, c^b, and γ^b to denote the corresponding parts in the objective function. We emphasize again that the cut pseudo base in use is fixed and contains $B = \lceil \log 2(n) \rceil$ many cuts.

Domain of $\boldsymbol{Y}$ We make sure that every $y_i^b \in \{-1, 1\}$. For the linear variables, we relax this constraint to $y_i^b \in [-1, 1]$, but in the SDP, we can require something stronger. By using the fact that $\left(y_i^b\right)^2 = 1$, we can add the constraint $Y_{ii}^b = 1$ for all $b \in [B], i \in [n]$. Formally, we do this by the SDP constraint $E_i \bullet Y^b = 1$ where E_i has a 1 on index (i, i) and 0s everywhere else.

Injectivity of the assignment We ensure that the assignment is injective, i.e., that no two different keys are assigned to the same spot. In terms of y variables, we require for all distinct i and j that y_i^b be different from y_j^b for at least one b. We have $y_i^b = y_j^b$ if and only if the corresponding entry in Y, namely Y_{ij}^b, is 1. Hence, we add the constraint

$$\sum_{b=1}^{B} Y_{ij}^b \leq B - 1 \Leftrightarrow \left(\sum_{b=1}^{B} \frac{1}{2} Y_{ij}^b + \frac{1}{2} Y_{ji}^b \right) \leq B - 1.$$

Note that in an integer optimal solution, the constraints above already ensure all the properties, we want to have. However, we have found that it is beneficial for the relaxed SDP to add the following constraint.

Zero row sums In the original formulation, injectivity implies that the number of keys assigned to one side of a cut is as large as the number of keys assigned to the opposite side. Recall that we are assuming $n = 2^k$, and we have a cut pseudo base. Hence, the implication above indeed holds. In terms of y variables, this can be modeled as the constraint

$$\sum_{j=1}^{n} y_j^b = 0$$

or as

$$\sum_{j=1}^{n} Y_{ij}^b = \sum_{j=1}^{n} y_i^b y_j^b = y_b^i \cdot \sum_{j=1}^{n} y_j^b \overset{!}{=} 0$$

in the SDP for an arbitrary fixed $i \in [n]$. Hence, taking the row sum of Y^b in this case yields the term we are looking for.

Total entry sum We have observed that we can condense the zero-sum-constraints to a single one by exploiting the positive semidefiniteness of Y.

Lemma 1. *Let $Y \in \mathbb{R}^{n\times n}$ be positive semidefinite. If $\mathbb{1}\mathbb{1}^T \bullet Y = 0$, then for any $i \in [n]$, it holds that $\sum_{j=1}^{n} Y_{ij}^b = 0$.*

Proof. Observe that
$$0 = \mathbb{1}\mathbb{1}^T \bullet Y = \mathbb{1}^T Y \mathbb{1}.$$
Since Y is positive semidefinite, $\mathbb{1}$ is an eigenvector of Y with eigenvalue 0, which implies that $Y\mathbb{1} = 0\mathbb{1} = \mathbf{0}$ and proves the claim.

Hence, instead of imposing n constraints for every single row of Y, we have shown that one constraint is enough to fix all row sums to 0.

2.4 Alternative objective functions for the SDP

In the previous subsection, we first fixed some cut b and then derived a lower bound on the minimum QAP objective value by minimizing an SDP relaxation. That is, we obtain a valid lower bound by solving an SDP with the objective function $C^b \bullet Y^b + (c^b)^T y^b + \gamma^b$, subject to the constraints mentioned above. However, considering only one cut of the pseudo base in the objective could be weak because costs could be evaded by charging them to the other cuts of the pseudo base that are not accounted for in the objective.

Averaging over the cut pseudo base Since the lower bound holds for arbitrary cuts b, it also holds for the average over all cuts in the cut pseudo base, i.e., the objective becomes
$$\frac{1}{B}\sum_{b=1}^{B} C^b \bullet Y^b + (c^b)^T y^b + \gamma^b.$$
There is no need to add further auxiliary variables or constraints that may harm the numeric stability of an SDP-solver.

Taking the maximum An even stronger lower bound is obtained by taking the maximum over the cuts of the pseudo base because the arithmetic mean never exceeds the maximum. The standard way to model the maximum over the cut pseudo base is to introduce a new linear variable - say z - and add $\log_2 n$ many constraints, ensuring that z is at least the cost of each cut in the pseudo base. However, the Mosek solver (v7.1.0.53) often stalled with this objective function, in contrast to the averaging objective.

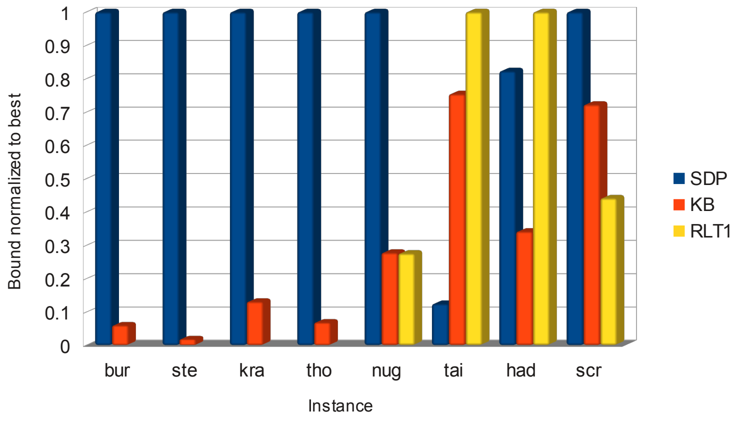

Figure 1: Averaged QAPLIB instances after one hour of computation

We are now ready to plug this SDP into a branch-and-bound framework of the QAP. Recall that we branch on the cut variables. However, as soon as we encounter an integral solution, there is a one-to-one-correspondence between the cut variables and the original assignment variables. This means, the cost of every incumbent is calculated with the original formulation. Hence, the framework will produce an optimal solution for any general QAP.

3 Evaluation

We compare our approach to two classical linearizations, the Kaufman-Broeckx linearization [21] and the first level of the Reformulation Linearization Technique (RLT1) [20]. We use the commercial state-of-the-art solver Gurobi (v6.5.1) [27] to solve the linearizations, and we use Mosek (v7.1.0.53) [26] as the SDP solver. All three approaches are embedded in a branch-and-bound framework. We will report the best known lower bound produced by running the branch-and-bound process for one hour.

We ran experiments on a compute server restricted to one Intel (R) Xeon (R) E5-2680 2.50GHz core and a limited amount of 8 GB RAM running Debian GNU/Linux 7 with kernel 3.18.27.1. The code was compiled with gcc version 4.7.2 using the `-O3` flag. The instances are taken from the QAPLIB homepage [11]. The names of the instances are formed by the name of the author (first three letters), the number of items, followed by a single letter identifier. The test instances cover a wide range of QAP applications including keyboard assignment, hospital layout and several further graph problems.

Figure 1 shows the average lower bound of the different approaches after one hour of computation time. We decided to average the lower bounds of a certain instance set because the single test cases within that set were similar and all approaches behaved consistently there. One can see that RLT1 performs quite well if we have enough computation power to compute bounds there (see `tai` or

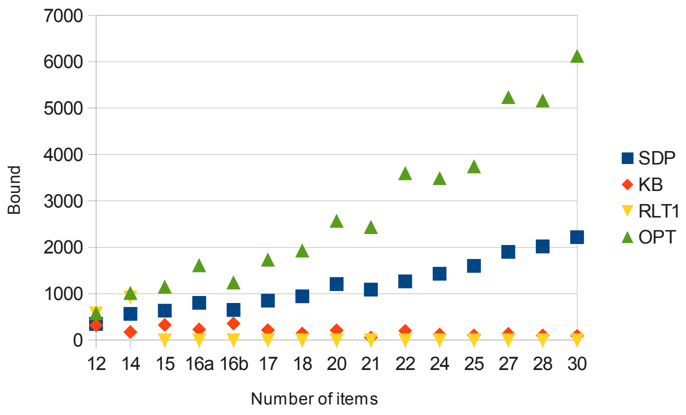

Figure 2: The Nugent instances with 12 to 30 items after one hour of computation, RLT1 fails to solve $n \geq 15$ within this time and resource limit.

`had`, for example). However, many test instances are too large for our computing resources to compute even the RLT1 root relaxation. In these *hard* cases, our approach outperforms both linear relaxations by several orders of magnitude.

The `nug` instances are a special instance set because this set contains test cases of increasing size. Therefore, the behavior of the three approaches varies throughout the different test cases and the average reported in Figure 1 cannot reflect the overall behavior for `nug`. To this end, we report a detailed description of the whole `nug` test set in Figure 2. This also shows how our framework scales with increasing n. At the beginning, for small n, RLT1 can solve the instances even to optimality, which meets our expectations. For instances of small size, the advantages of our approach in efficiency are just too small to make up for the loss of precision caused by the distance approximations. The trend changes as soon as n grows above 16. The RLT1 formulations are already too large to solve the root relaxation after one hour of computation with a single thread and the bounds of the Kaufman-Broeckx relaxation are lower than ours. This confirms the use-case that we proposed in the introduction of this paper.

4 Future Work

We plan to work on several heuristics to produce better incumbents during the branching, e.g., using randomized rounding. The current version fully focuses on producing good lower bounds. Adding good primal heuristics could improve the framework further. Furthermore, it is still open whether our formulation can be improved by further cutting planes. An idea is to add triangle inequalities. Since these are potentially many additional constraints, we might consider the idea of dynamic constraint activation, which does [28], for example.

Moreover, we plan to tackle some numerical stability issues in our framework. Although our problem satisfies Slater's condition, it is clear that the primal SDP never contains an interior point (because the balanced constraints force at least one eigenvalue to 0). Recent approaches (e.g., [22]) consider this problem and reformulate the SDP they use such that both primal and dual problems are strictly feasible in order to improve the numerical stability. We will investigate whether it is possible to adapt this idea to our framework.

5 Acknowledgments and Supplementary Material

This research was funded by the Max Planck Center for Visual Computing and Communication (www.mpc-vcc.org).

Additional information and detailed evaluation data are released on the project homepage at http://resources.mpi-inf.mpg.de/qap/.

References

1. Birkhoff, D.: Tres observaciones sobre el algebra lineal. Universidad Nacional de Tucuman Revista , Serie A **5** (1946) 147–151
2. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly **2** (1955) 83–97
3. Duan, R., Su, H.H.: A scaling algorithm for maximum weight matching in bipartite graphs. In: Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '12, SIAM (2012) 1413–1424
4. Koopmans, T., Beckmann, M.J.: Assignment problems and the location of economic activities. Cowles Foundation Discussion Papers 4, Cowles Foundation for Research in Economics, Yale University (1955)
5. Nugent, C., Vollman, T., Ruml, J.: An experimental comparison of techniques for the assignment of facilities to locations. Operations Research **16** (1968) 150–173
6. Burkard, R., Offermann, J.: Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme. Zeitschrift für Operations Research **21** (1977) 121–132
7. Rainer E. Burkard, Eranda Çela, Panos M. Pardalos and Leonidas S. Pitsoulis: The Quadratic Assignment Problem (1998)
8. Steinberg, L.: The Backboard Wiring Problem: A Placement Algorithm. SIAM Review **3** (1961) 37–50
9. Krarup, J., Pruzan, P.M.: Computer-aided layout design. In: Mathematical Programming in Use. Springer Berlin Heidelberg, Berlin, Heidelberg (1978) 75–94
10. Elshafei, A.N.: Hospital layout as a quadratic assignment problem. Operational Research Quarterly (1970-1977) **28** (1977) 167–179
11. Burkard, R.E., Karisch, S.E., Rendl, F.: Qaplib - a quadratic assignment problem-library. J. of Global Optimization **10** (1997) 391–403
12. Anstreicher, K., Brixius, N., Goux, J.P., Linderoth, J.: Solving large quadratic assignment problems on computational grids. Mathematical Programming **91** (2014) 563–588
13. Queyranne, M.: Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems. Operations Research Letters **4** (1986) 231 – 234

14. Pardalos, P.M., Rendl, F., Wolkowicz, H.: The quadratic assignment problem: A survey and recent developments. In: In Proceedings of the DIMACS Workshop on Quadratic Assignment Problems, volume 16 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society (1994) 1–42
15. Commander, C.W.: A survey of the quadratic assignment problem, with applications. Morehead Electronic Journal of Applicable Mathematics **4** (2005) MATH–2005–01
16. Loiola, E.M., de Abreu, N.M.M., Boaventura-Netto, P.O., Hahn, P., Querido, T.: A survey for the quadratic assignment problem. European Journal of Operational Research **176** (2007) 657 – 690
17. Gilmore, P.C.: Optimal and suboptimal algorithms for the quadratic assignment problem. SIAM J. Appl. Math. **10** (1962) 305–313
18. Lawler, E.L.: The quadratic assignment problem. Management Science **9** (1963) 586–599
19. Li, Y., Pardalos, P.M., Ramakrishnan, K.G., Resende, M.G.C.: Lower bounds for the quadratic assignment problem. Annals of Operations Research **50** (1994) 387–410
20. Frieze, A., Yadegar, J.: On the quadratic assignment problem. Discrete Applied Mathematics **5** (1983) 89 – 98
21. Kaufman, L., Broeckx, F.: An algorithm for the quadratic assignment problem using benders' decomposition. European Journal of Operational Research **2** (1978) 204–211
22. Zhao, Q., Karisch, S.E., Rendl, F., Wolkowicz, H.: Semidefinite programming relaxations for the quadratic assignment problem. Journal of Combinatorial Optimization **2** (1998) 71–109
23. Povh, J., Rendl, F.: Copositive and semidefinite relaxations of the quadratic assignment problem. Discret. Optim. **6** (2009) 231–241
24. Peng, J., Mittelmann, H., Li, X.: A new relaxation framework for quadratic assignment problems based on matrix splitting. Mathematical Programming Computation **2** (2010) 59–77
25. Wolsey, L.A.: Integer programming. Wiley-Interscience series in discrete mathematics and optimization. J. Wiley & sons, New York (N.Y.), Chichester, Weinheim (1998) A Wiley-Interscience publication.
26. MOSEK ApS: The MOSEK C optimizer API manual Version 7.1 (Revision 52). (2016)
27. Gurobi Optimization, I.: Gurobi optimizer reference manual (2016)
28. Rendl, F., Rinaldi, G., Wiegele, A.: Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. Mathematical Programming **121** (2008) 307–335