Dynamic Sparsification for Quadratic Assignment Problems

Maximilian John and Andreas Karrenbauer

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

Abstract We present a framework for optimizing sparse quadratic assignment problems. We propose an iterative algorithm that dynamically generates the quadratic part of the assignment problem and, thus, solves a sparsified linearization of the original problem in every iteration. This procedure results in a hierarchy of lower bounds and, in addition, provides heuristic primal solutions in every iteration. This framework was motivated by the task of the French government to design the French keyboard standard, which included solving sparse quadratic assignment problems with over 100 special characters; a size not feasible for many commonly used approaches. Designing a new standard often involves multiple stakeholders having conflicting opinions and, hence, no agreement on a single well-defined objective function to be used for an extensive one-shot optimization. Since the process of designing the standard is highly interactive, it demands rapid prototyping, e.g., quick primal solutions, on-the-fly evaluation of manual changes, and prompt assessments of solution quality. Particularly concerning the latter aspect, our algorithm is able to provide high-quality lower bounds for these problems within only a few minutes.

Keywords: quadratic assignment · integer programming · linearization · keyboard optimization

1 Introduction

Assignment problems aim at finding the cheapest one-to-one correspondence between n items and locations. Already in 1946, Birkhoff [4] showed that the optimal assignment can be found in $\mathcal{O}(n^3)$ time if the objective function is linear. However, linear objective functions cannot capture pairwise dependencies between variables. Koopmans and Beckmann [16] investigated a variant with quadratic terms in the cost function. This quadratic optimization problem includes several practical applications, such as the keyboard layout problem [6], the facility location problem [19], the traveling salesman problem [5], and many others.

As expected, great modeling power comes with increased hardness; there are problems of only $n = 30$ items that cannot be solved to optimality in reasonable time. From a complexity theoretical point of view, Queyranne [23] showed that the quadratic assignment problem (QAP) is NP-hard to approximate within any

constant factor, even if the quadratic cost can be factorized to a symmetric block diagonal matrix and a distance matrix describing a line metric.

To cope with the hardness, researchers have proposed many strategies over the last decades. Many of them are based on linearizations, e.g., the classical results by Gilmore [11] and Lawler [17], and by Kaufman and Broeckx [15]. These linearizations can be considered light, meaning that their space requirements are linear in the input size and the corresponding relaxations can be solved quickly, e.g., in a branch-&-bound framework. Moreover, the latter approach seems to be amenable for primal heuristics of state-of-the-art MIP solvers to compute good incumbents. However, their lower bounds deteriorate quickly with increasing input size, which negatively impacts the performance of branch-&-bound. More recently, new improved linearizations have been developed by Xia and Yuan [25] and Zhang [26], who combine the ideas of the light-weight approaches mentioned above.

On the contrary, the formulations by Frieze and Yadegar [10] and by Adams and Johnson [1] compute very strong lower bounds at the expense of $\mathcal{O}(n^4)$ additional variables. Both approaches yield equivalent formulations for the QAP and are commonly referred to as RLT1 formulations, a more general concept proposed by Sherali and Adams [24]. Huber and Riedl showed [13] that the Adams-Johnson formulation dominates the one of Xia and Yuan. For many instances of practical size and especially in our scenario, RLT1 and other similar more powerful approaches could not produce any results within the given resources, which we expected, due to the sheer size of the problem.

Recently, semidefinite programming relaxations for QAPs [14,22,27] have become more popular. Peng et al. [20] showed that these approaches indeed often produce good lower bounds for the QAP.

1.1 Keyboard optimization as assignment problems

Already in the 70s, Pollatschek [21] as well as Burkard and Offermann [6] proposed to optimize keyboard layouts as a quadratic assignment problem. They consider the assignment problem with mixed linear and quadratic terms in the objective function. Formally, let the n characters and keys be numbered from 1 to n. We refer to the set of characters as $[n] := \{1, \dots, n\}$. Furthermore, let $x_{ik} \in \{0, 1\}$ denote the decision of whether or not to assign character i to key k. With c_{ik} being the linear and $q_{ijk\ell}$ the quadratic assignment cost, we obtain the following quadratic program.

$$\begin{aligned} \min \quad & \sum_{i,k=1}^{n} c_{ik} x_{ik} + \sum_{i,j,k,\ell=1}^{n} q_{ijk\ell} x_{ik} x_{j\ell} & \\ \text{subject to} \quad & \sum_{i=1}^{n} x_{ik} = 1 & \forall k \in [n] \\ & \sum_{k=1}^{n} x_{ik} = 1 & \forall i \in [n] \\ & x_{ik} \in \{0, 1\} & \forall i, k \in [n] \end{aligned} \tag{1}$$

The term $q_{ijk\ell}$ describes the cost of simultaneously assigning character i and j to the key k and ℓ, respectively. In keyboard problems, this term usually factors into $q_{ijk\ell} = p_{ij} \cdot d_{k\ell}$, where p_{ij} denotes the empirical probability of typing letter j after letter i, and $d_{k\ell}$ is the time between pressing the key slots k and ℓ.

Typically, integer linear programs are relaxed by dropping the integrality constraints of the variables. In this case, however, as the assignment polytope is well-understood, we can quickly solve linear assignment problems with over a million variables [18]. The hardness of QAPs comes, therefore, not from the polytope, but from the quadratic terms of the objective function. It is possible to exploit the structure of the objective so that special cases become more tractable, in some cases polynomial approximation algorithms have been developed [3]. In this work, we define a relaxation of the QAP, too. While also keeping the integrality constraints untouched, we exploit the sparse nature of the quadratic objective function and modify the quadratic terms to obtain upper and lower bounds for the original problem.

1.2 Designing the French keyboard standard

In 2015, the French Ministry of Culture discussed the concerns about not having an official French keyboard standard [7]. The commonly used *AZERTY* layout did not permit typing frequent special characters like À, œ, etc. One year later, AFNOR, the French national organization for standardization, was designed the task of designing the new standard [8], which should support all missing special characters that are used in the French language. Two major options were discussed: optimizing the whole keyboard from scratch, or keeping the most frequent characters (*A-Z*) fixed and only optimizing the addition of over 100 *special characters*, the latter of which would maintain familiarity and facilitate learnability. We participated in the endeavor to achieve the second option.

The process of defining the standard consisted of several rounds of gathering data (details in [9]), modeling the problem as a QAP, finding a (near-)optimal keyboard, presenting it to an official committee who expressed further wishes for the objective, modifying the weighting of its components, and adding or removing certain characters. Eventually, the objective function stabilized as a conic combination of four different measures: **performance** – special characters that are often used in combination with fixed characters should be close together in order to minimize the time to type; **ergonomics** – frequently used special characters should be quite central on the keyboard to avoid unhealthy stretches of the fingers; **intuitiveness** – special characters should be placed close to similar fixed characters and other special characters to simplify finding them on the keyboard; **familiarity** – frequent special characters should be placed close to their position in the original AZERTY keyboard if this character was already positioned there. Note that every interaction between special characters and fixed characters can be modeled as linear expressions because we are not allowed to change the position of these fixed characters, therefore keeping their impact to the objective function constant. Hence, 3 out of 4 of these measures describe a linear objective function. The quadratic part of the objective function models

Figure 1: The new French keyboard standard (NF Z71-300). Special characters (blue) and diacritic marks (red) were added to the old AZERTY layout. More information about the new standard can be found on https://norme-azerty.fr

the similarity of two special characters, which is part of the intuitiveness measurement. Since there is only a restricted number of similar special characters, e.g., é and è, this explains the sparsity of the quadratic objective function.

After a first consensus had been found, a public inquiry organized by AFNOR provided feedback on the proposed design. Not only after the public inquiry, but also after every iteration of this feedback loop, the underlying model was updated and new solutions were heuristically computed; which our algorithm then showed to be near-optimal. Why is this last step important? In contrast to one-shot optimization with a single well-defined objective, the committee, which consists of multiple stakeholders with different interests and opinions, discusses several solutions, evaluates the impact of manual changes on different parts of the objective, and alters the optimization model to find a compromise. Deciding model changes based on very sub-optimal solutions is pointless, since the observed solution might not properly represent the current model. Therefore, it is not only important to find near-optimal solutions, but also to have a sharp picture of the their quality, allowing for a well-founded discussion and decision process. Additionally, it is important that such solutions to updated models and corresponding bounds can be computed as fast as possible, ideally even in real-time.

Finally, the expert committee agreed on a layout for the new French keyboard standard [2], which is depicted in Figure 1 and was launched on 2 April 2019.[1]

1.3 Our contribution

The goal of our framework is to utilize the power of the RLT1 approach while avoiding the computational overhead. We present an algorithm that dynami-

[1] https://normalisation.afnor.org/actualites/faq-clavier-francais/ – retr. 2019-04-03

cally generates the quadratic terms of the QAP and leads to a hierarchy of lower bounds and heuristic primal solutions at the same time. In contrast to a classic column-generation approach, our algorithm guarantees a sequence of non-decreasing lower bounds in every step instead of non-increasing upper bounds. This iterative framework produces a $(1+\varepsilon)$-approximation[2] for the QAP for any $\varepsilon \geq 0$. We show the success of our framework during the design process of the new French keyboard standard. The lower bounds computed by our algorithm showed very small optimality gaps within a few minutes for sparse QAPs with over 100 items and 130 locations. All examples shown in this paper are real-world instances created during this standardization process. Hence, our tool is usable to provide almost real-time feedback with very limited resources, e.g., a laptop.

2 Algorithm

Linear relaxations for quadratic programs have been extensively studied over the last decades and are still a good starting point for many new ideas. However, the disadvantage of standalone linear relaxations is either high space complexity or an inefficient bound generation. We want to overcome this issue for QAPs with sparse quadratic objectives.

Let $\mathcal{S} \subseteq [n]^4$ be a set of indices. We define the following subproblem of (1).

$$\begin{aligned}
\min \quad & \sum_{i,k=1}^{n} c_{ik} x_{ik} + \sum_{(i,j,k,\ell)\in\mathcal{S}} q_{ijk\ell} y_{ijk\ell} && && \text{(2a)}\\
\text{subject to} \quad & \sum_{i=1}^{n} x_{ik} = 1 && \forall k \in [n] &\\
& \sum_{k=1}^{n} x_{ik} = 1 && \forall i \in [n] &\\
& \sum_{j:(i,j,k,\ell)\in\mathcal{S}} y_{ijk\ell} \leq x_{ik} && \forall i,k,\ell \in [n] & \text{(2b)}\\
& \sum_{\ell:(i,j,k,\ell)\in\mathcal{S}} y_{ijk\ell} \leq x_{ik} && \forall i,j,k \in [n] & \text{(2c)}\\
& \sum_{i:(i,j,k,\ell)\in\mathcal{S}} y_{ijk\ell} \leq x_{j\ell} && \forall j,k,\ell \in [n] & \text{(2d)}\\
& \sum_{k:(i,j,k,\ell)\in\mathcal{S}} y_{ijk\ell} \leq x_{j\ell} && \forall i,j,\ell \in [n] & \text{(2e)}\\
& x_{ik} + x_{j\ell} \leq 1 + y_{ijk\ell} && \forall (i,j,k,\ell) \in \mathcal{S} & \text{(2f)}\\
& y_{ijk\ell} \in [0,1] && \forall (i,j,k,\ell) \in \mathcal{S} &\\
& x_{ik} \in \{0,1\} && \forall i,k \in [n] &
\end{aligned}$$

[2] We give no polynomial time guarantee. A PTAS would imply P = NP.

Note that it is feasible to add symmetry constraints for the y-variables of the form $y_{ijk\ell} = y_{ji\ell k}$ inspired by the Adams-Johnson formulation because they simulate the commutative multiplication of x_{ik} and $x_{j\ell}$, however, we could not observe any performance gain, and thus, omit them.

We first show that the proposed formulation is exact in the boundary case $\mathcal{S} = [n]^4$.

Lemma 1. *Let $\mathcal{S} = [n]^4$ and let $z^{(1)} = x^{(1)}$, $z^{(2)} = (x^{(2)}, y^{(2)})$ be optimal solutions of* (1) *and* (2)*, respectively.*

Then $cost(z^{(1)}) = cost(z^{(2)})$.

Proof. Let $(i, j, k, \ell) \in [n]^4$ and consider the linear inequalities (2b)-(2f). If one of $x^{(2)}_{ik}$ and $x^{(2)}_{j\ell}$ is 0, then at least one of the inequalities (2b) to (2e) forces $y^{(2)}_{ijk\ell}$ to 0. On the other hand, if both $x^{(2)}_{ik} = x^{(2)}_{j\ell} = 1$, constraint (2f) sets $y^{(2)}_{ijk\ell}$ to 1. Therefore, and because $x^{(2)}$ is a binary variable, we can interpret $y^{(2)}_{ijk\ell}$ as the product $x^{(2)}_{ik} \cdot x^{(2)}_{j\ell}$.

Since $\mathcal{S} = [n]^4$, this observation holds for all variables and the formulations (1) and (2) coincide. □

Despite this result, we emphasize that using $\mathcal{S} = [n]^4$ leads to an intractable problem size for most practical input instances, e.g., more than 100 000 000 variables in our application. Even for sparse problems, reducing $\mathcal{S}$ to all the indices with nonzero contribution to the quadratic objective term may not suffice as, e.g., still about 2 000 000 variables remain in our case. To overcome this issue, we select an increasing sequence of subsets $\mathcal{S}$, with each subset being significantly smaller than $[n]^4$. Lemma 2 explains why it is beneficial to do so.

Lemma 2. *Let $\mathcal{S} \subset [n]^4$ and z^* be the optimal solution of* (2)*. Then $cost(z^*)$ is a lower bound for* (1)*.*

Proof. Let $(P, c, q), (P', c', q')$ be the polytopes and objective functions of (2) defined over $\mathcal{S}$ and $[n]^4$, respectively. Clearly, the set of constraints of P form a subset of the constraints of P'. Hence, every feasible solution in P' is also feasible in P, i.e., $P' \subseteq P$.

Setting $q_{ijk\ell} = 0$ for all $(i, j, k, \ell) \notin \mathcal{S}$, we can write (2a) as

$$\sum_{i,k=1}^{n} c_{ik} x_{ik} + \sum_{(i,j,k,\ell) \in [n]^4} q_{ijk\ell} y_{ijk\ell}.$$

Since all terms in the objective functions are assumed to be non-negative, it holds for every $(i, j, k, \ell) \in [n]^4$ that $q_{ijk\ell} \leq q'_{ijk\ell}$, which concludes the proof. □

This lemma shows that dynamically increasing $\mathcal{S}$ yields a hierarchy of integer linear programs with increasing bounds for the original QAP. The proposed iterative algorithm later in this section is a natural consequence of Lemma 2. It remains to show how to initialize and update the set $\mathcal{S}$. We remark here that

it is advisable to solve the integer linear programs close to optimality instead of considering their linear programming relaxations. Although dropping the integrality constraints drastically reduces the computation time with growing $\mathcal{S}$, the resulting lower bounds have shown to be significantly worse than the ones obtained by solving the integral versions for the same amount of time.

We now present two variants of the algorithm, which differ only in the procedure on how to grow $\mathcal{S}$.

Variant 1: conservative growth First, we choose an arbitrary $\varepsilon \geq 0$. We will show later that the algorithm then produces a $(1+\varepsilon)$-approximation of the optimal assignment. Note, however, that our algorithm allows to choose $\varepsilon = 0$, then computing an optimal solution. Assume that for a given index set $\mathcal{S}$, we computed an optimal binary solution (x^*, y^*) with objective value V. We build the candidate set

$$C := \left\{(i,j,k,\ell) \notin \mathcal{S} : x^*_{ik} = x^*_{j\ell} = 1 \text{ and } q_{ijk\ell} > 0\right\} \tag{3}$$

and sort C in an ascending order with respect to $q_{ijk\ell}$. Note that $|C| \leq n^2$. Formally, we define the function $\pi : [|C|] \to [n]^4$ such that for every $i < j \in \{1, \ldots, |C|\}$, it holds $q_{\pi(i)} \leq q_{\pi(j)}$. Let s be the index that satisfies the following equation.

$$s = \max\left\{t \in \{0, \ldots, |C|\} : \sum_{\alpha=1}^{t} q_{\pi(\alpha)} \leq \varepsilon \cdot V\right\} \tag{4}$$

Intuitively, we skip the s smallest positive cost values that sum up to a certain threshold and add the rest of the indices to our active set $\mathcal{S}$.

The complete algorithm is presented in Algorithm 1. Theorem 1 shows that the update step eventually yields a $(1+\varepsilon)$-approximation.

Input : number of items/locations n, linear cost c, quadratic cost q, precision parameter ε
Result: Optimal assignment or upper/lower bound if aborted

1 $\mathcal{S} \leftarrow \emptyset$;
2 **do**
3 $\quad (x^*, y^*) \leftarrow$ opt. sol. of (2) with $\mathcal{S}$;
4 $\quad V \leftarrow$ evaluate x^* at (1);
5 $\quad C, \pi, s$ as in equations (3)-(4);
6 $\quad \mathcal{S} \leftarrow \mathcal{S} \cup \{\pi(i)\}_{i=s+1}^{|C|}$;
7 **while** $\mathcal{S}$ *changed in line 6*;
8 **return** x^*;

Algorithm 1: The complete algorithm (conservative version)

Theorem 1. *Let $\varepsilon \geq 0$. If line 6 of Algorithm 1 does not add any index to $\mathcal{S}$, then the x-part of the current solution (x^*, y^*) is $(1+\varepsilon)$-optimal for problem (1).*

Proof. Since $C \cap \mathcal{S} = \emptyset$ by definition, the only reason why $\mathcal{S}$ did not change is that $s = |C|$. In particular, this means that

$$\sum_{\alpha \in C} q_\alpha \leq \varepsilon \cdot cost(x^*, y^*)$$

Let $\tilde{x}$ be the optimal solution of (1). We evaluate (x^*, y^*) on the complete objective function of the QAP and interpret $y^*_{ijk\ell} = x^*_{ik} x^*_{k\ell}$, which is a valid assumption already shown in the proof of Lemma 1. Then, we obtain an upper bound for $\tilde{x}$.

$$\begin{aligned} OPT &= \sum_{i,k=1}^{n} c_{ik}\tilde{x}_{ik} + \sum_{i,j,k,\ell=1}^{n} q_{ijk\ell}\tilde{x}_{ik}\tilde{x}_{j\ell} \\ &\leq \sum_{i,k=1}^{n} c_{ik}x^*_{ik} + \sum_{(i,j,k,\ell)\in\mathcal{S}} q_{ijk\ell}x^*_{ik}x^*_{j\ell} + \sum_{(i,j,k,\ell)\notin\mathcal{S}} q_{ijk\ell}x^*_{ik}x^*_{j\ell} \\ &= \sum_{i,k=1}^{n} c_{ik}x^*_{ik} + \sum_{(i,j,k,\ell)\in\mathcal{S}} q_{ijk\ell}x^*_{ik}x^*_{j\ell} + \sum_{(i,j,k,\ell)\in C} q_{ijk\ell}x^*_{ik}x^*_{j\ell} \\ &\leq cost(x^*, y^*) + \varepsilon cost(x^*, y^*) = (1+\varepsilon) cost(x^*, y^*) \end{aligned}$$

□

Variant 2: progressive growth We change the definition of the candidate set C in Equation (3) to

$$C' := \left\{(i,j,k,\ell) \notin \mathcal{S} : x^*_{ik} = 1 \vee x^*_{j\ell} = 1 \text{ and } q_{ijk\ell} > 0\right\}. \tag{5}$$

This means we consider a tuple as a candidate if at least one of the corresponding x-variables were set to 1 in the previous optimal solution (instead of requiring both variables to be 1). The rest of the algorithm remains the same. In this second variant, $|C'| \leq n^3$, i.e., we potentially add more terms to the model. This can improve the evolution of lower bounds because we consider a more substantial portion of the model more quickly. As a trade-off, we potentially add more irrelevant terms than the conservative variant and, additionally, we could quickly arrive at a model of a size that exceeds the resources of the computer used to run the algorithm. Note that Theorem 1 also holds for this variant of the algorithm, the proof is analogue to the proof shown above with the extra information that $C \subset C'$.

Variant 3: Hybrid strategy To achieve a balance between the fast evolution of lower bounds in variant 2 and the moderate growth of model size in variant 1, we propose the hybrid strategy that kick-starts with the progressive variant 2 and switches to the conservative variant 1 before the model size grows too large. The evaluation shows that this strategy is indeed superior to both standalone variants. For all instances, there is a critical point where the amount of generated quadratic terms would grow so large that an MIP solver cannot compute the integer optimal solution within a reasonable time. Therefore, we switch to the conservative variant at this critical point, which grows the model more slowly while still steadily improving the lower bound.

3 Evaluation

We applied our algorithm within several stages of the French keyboard standardization process. The instances consist of over 100 special characters and 130 keys (in order to achieve the classic QAP formulation, one can generate dummy characters symbolizing that a key is left empty), and the objective function consists of a conic combination of a sparse quadratic and dense linear cost terms. The quadratic term can be factorized into a sparse matrix F, which describes the association score (similarity) of two different special characters, and a dense matrix D, describing the distances between two key slots. The weight of the quadratic part ranges between 30% and 50%. Additionally, some instances fix few characters like punctuation symbols to fixed slots or require that the capital versions of special characters are placed on the shifted slot of the same letter (e.g., È is placed on the shifted slot of è) whereas other instances also allow them to be on the Alt-Shift or Alt version of this slot.

We evaluated the instances on a single Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz processor core with 16GB of RAM. We compare our algorithm against the formulation of Xia and Yuan [25] as a state-of-the-art lightweight linearization for the QAP. As already mentioned before, stronger formulations like RLT1 could not compute any lower bound within the given resources, often because the model size already exceeded the available RAM. We use Gurobi version 8.1 [12] as the underlying solver for both approaches.

We first discuss the impact of the variant choice on one example instance. More specifically, we test the hybrid strategy and the effect of the switch from

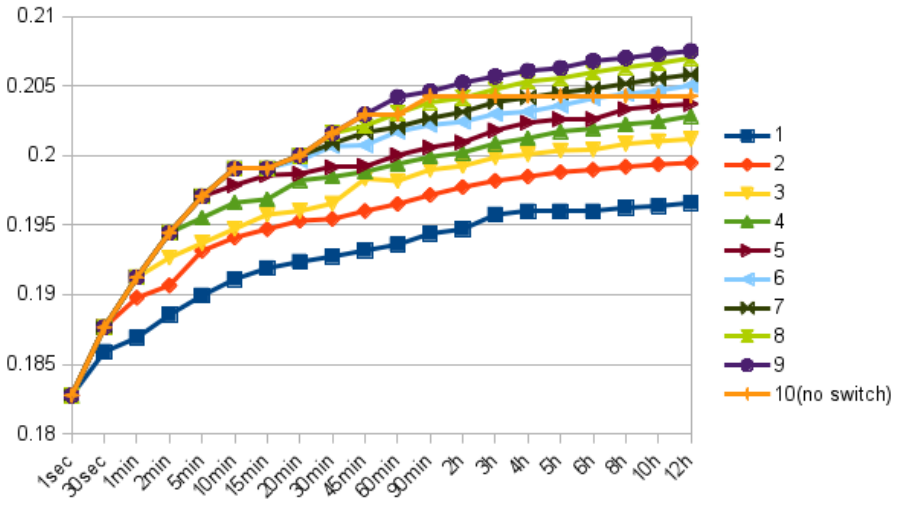

Figure 2: The evolution of the lower bound when switching variants for instance N50s. The numbers in the legend describe the iteration at which the switch was triggered.

progressive to conservative at different iterations τ. Figure 2 shows the evolution of lower bounds for $\tau = 1, \ldots, 10$. Since naturally the bound evolves faster during the first seconds and minutes, the graph shows a more detailed view on the evolution within this first period. Setting $\tau = 1$ leads to using the conservative variant from the beginning while setting $\tau = 10$ implies that the strategy switch does not occur within the given time window of 12 hours because the model of the last iteration is already too large to be solved efficiently. We observe that as long as the model size is moderately low, the progressive variant achieves better results at every time stamp. However, after roughly 45 minutes, the model size for this variant already grows notably large so that the next iterations takes quite a long time. After yet another size increase, the ILP solver could not compute an optimal solution within the remaining 10 hours. Note that depending on the available resources (time limit and hardware) as well as the particular instance (dimension and sparsity), the critical point at which a switch from the progressive to the conservative variant is valuable varies. Since all our instances are of similar size and sparsity, the critical point for this evaluation is at the 9th iteration.

Figure 3 compares the evolution of the lower bounds of our algorithm using only variant 2, switching after 9 iterations, and the formulation of Xia and Yuan within a total time period of 12 hours for the same example instance. It is important to note that the setup time for the Xia-Yuan formulation is around 25 minutes for every instance because over 10 000 linear assignment problems are solved beforehand. Therefore, the first bound for the original QAP is only produced after 25 minutes.

To avoid visual clutter in the following figures, we only depict the results of the hybrid algorithm switching at the 9th iteration. The lower and upper bounds for all QAP instances are shown in Figure 4. We ran our hybrid algorithm for

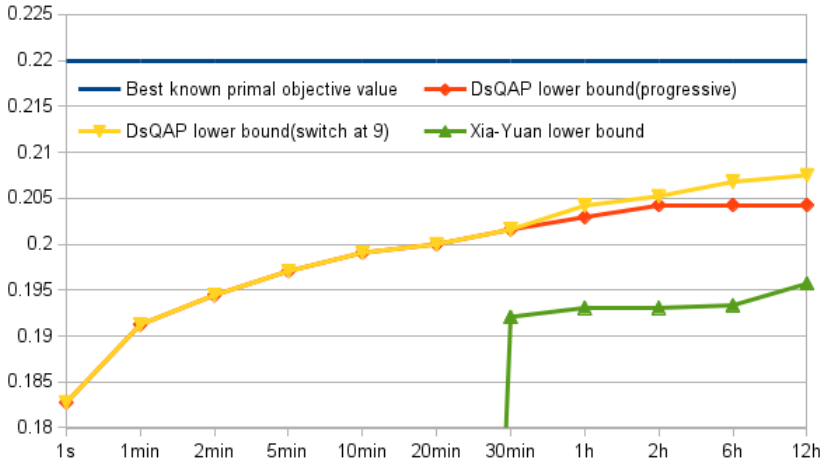

Figure 3: The evolution of the lower bounds within 12 hours of computation time for instance N50s.

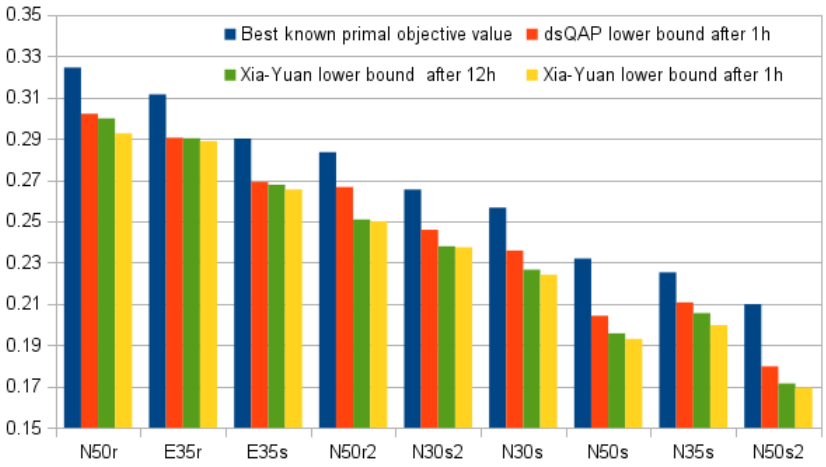

Figure 4: Lower and upper bounds for the QAP instances

one hour and compare it against the formulation of Xia and Yuan after one and 12 hours of computation time.

The naming of the test instances is as follows: the first letter describes the set of additional constraints (N for no additional constraints, and E for fixed punctuation symbols and the fixed symbols è, é, ê, à, and €). The number in the middle describes the weight (in percent) of the quadratic term in the objective function, and the following letter describes if the capitalized letter of a special character has to be placed on the shifted slot (s) or on any alternative of this slot (r). Note that almost every instance uses a slightly different set of characters because this set constantly changed in committee meetings. The full description of all the different character sets and further details about the data gathering is beyond the scope of this paper and can be found in [9]. Therefore, it occurs that two instances are equally named although they slightly differ in the character set used. In this case, one of the instance names ends with 2 for better differentiation.

We can see that within one hour, we outperform the formulation of Xia and Yuan for every instance independent of its time limit being one hour or 12 hours. Although we slightly improved the lower bounds of all instances, this is not the true benefit of our framework. What we really want to emphasize here is how fast we achieve high-quality lower bounds, which is especially important in the practical application of our algorithm. In this highly interactive environment with countless model updates and changes, receiving valuable feedback of an optimization method after only several minutes can greatly improve the dynamics of an expert committee that discusses different proposals and has to decide the next steps towards a final keyboard standard.

We measure the time our algorithm needs to exceed the lower bounds that the Xia-Yuan formulation produces after 1 hour and 12 hours, respectively. Figure 5 shows that we achieve this goal within several minutes for all instances. In the

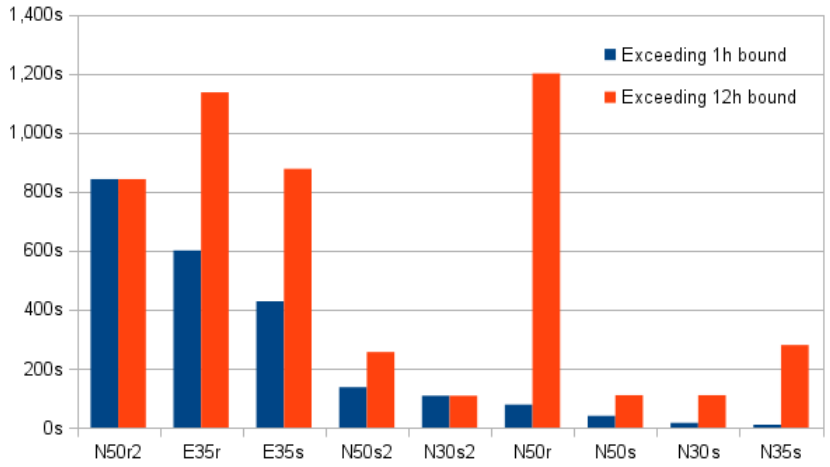

Figure 5: The time we need to exceed the bounds of Xia and Yuan after $1h$ and $12h$ (in seconds)

worst case, it takes 20 minutes to exceed the 12 hours bound of Xia-Yuan. Hence, for every instance, we achieve superior lower bounds within the setup time of 25 minutes that is needed for the creation of the Xia-Yuan linearization.

3.1 Robustness Analysis

To analyze the robustness of our approach, we vary the nonzero values of the quadratic cost matrix with additive noise generated by a normal distribution with 0 mean and standard deviation σ.

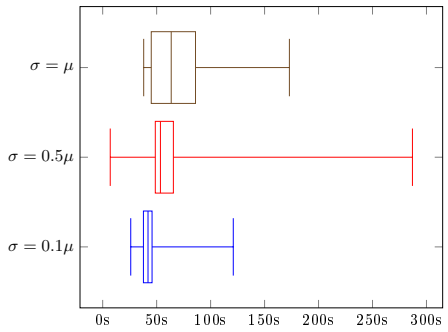

Figure 6: Boxplots of the time (in seconds) until our approach exceeded the 12 hours Xia-Yuan bound

Recall that the quadratic matrix Q is the Kronecker product of the dense matrix D containing the distances between the keys and the sparse matrix F encoding the similarity between the special characters. We only add noise to the entries in F while keeping its entries non-negative. More specifically, consider $f_{ij} > 0$ and $\delta_{ij} \sim N(0, \sigma)$, then we set $f'_{ij} = f_{ij} + \delta_{ij}$ if $f'_{ij} > 0$, otherwise we recompute δ_{ij}. Let μ be the average value of all nonzero entries in the association matrix A, then we set σ to 10%, 50%, and 100% of μ. For this evaluation, we use the instance N35s as a base instance and generate 20 randomly variated instances for each of the three variance values.

Figure 6 shows the boxplots of the time (in seconds) our approach needed to exceed the bound that the Xia-Yuan formulation achieves after 12 hours. In every of the 60 instances in total, we exceeded said bound after at most five minutes. Note that the Xia-Yuan formulation has a setup time for around 25 minutes for instances of this size. This means we can consistently produce high quality bounds during the setup time of the competing approach.

Moreover, Figure 7 shows the lower and upper bounds for the 20 runs each with $\sigma \in \{0.1\mu, 0.5\mu, \mu\}$, respectively. We observe that the results of these randomized instances are very consistent with the results of the original evaluation, independent of the variance.

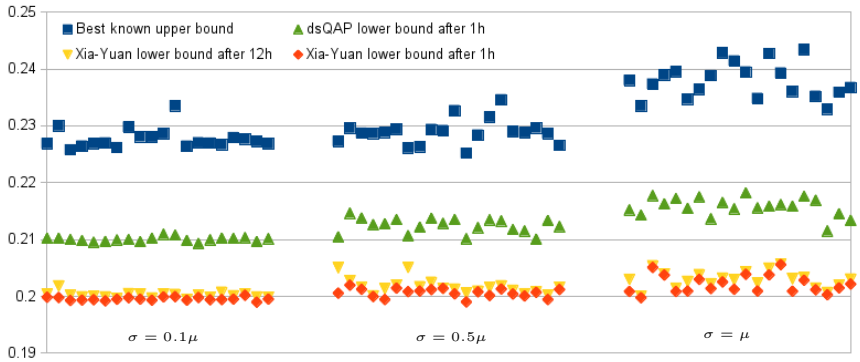

Figure 7: Bounds for 60 randomly variated instances (20 each) with variance σ

4 Conclusion

We presented a lightweight framework for sparse quadratic assignment problems that combines powerful linearization techniques and ideas from column-generation. It is lightweight in a sense that it can generate good bounds for sparse QAPs of huge size (over 100 items) on a normal laptop. Our algorithm was used in the process of defining the new French keyboard standard. The evaluation, which is based on real data gathered during this standardization process,

showed that we can compete with state-of-the-art linearization techniques. We showed that we can produce high quality lower bounds within several minutes, which serves the purpose of almost real-time feedback in such a dynamic interactive optimization process.

References

1. Adams, W., Johnson, T.: Improved Linear Programming-Based Lower Bounds for the Quadratic Assignment Problem. DIMACS 512 Series in Discrete Mathematics and Theoretical Computer Science **16** (08 1994). https://doi.org/10.1090/dimacs/016/02
2. AFNOR: Interfaces utilisateurs - Dispositions de clavier bureautique français, NF Z71-300 Avril 2019. La Plaine Saint-Denis: AFNOR, Version de 2019-04-P, 85 p.
3. Arkin, E.M., Hassin, R., Sviridenko, M.: Approximating the maximum quadratic assignment problem. Information Processing Letters **77**(1), 13 – 16 (2001). https://doi.org/10.1016/S0020-0190(00)00151-4
4. Birkhoff, D.: Tres observaciones sobre el algebra lineal. Universidad Nacional de Tucuman Revista , Serie A **5**, 147–151 (1946)
5. Burkard, R.E., Çela, E., Pardalos, P.M., Pitsoulis, L.S.: The Quadratic Assignment Problem, pp. 1713–1809. Springer US, Boston, MA (1998). https://doi.org/10.1007/978-1-4613-0303-9_27
6. Burkard, R., Offermann, J.: Entwurf von Schreibmaschinentastaturen mittels quadratischer Zuordnungsprobleme. Zeitschrift für Operations Research **21**, 121–132 (1977)
7. DGLFLF: Rapport au Parlement sur l'emploi de la langue française. Government Report (2015), http://www.culture.gouv.fr/Thematiques/Langue-francaise-et-langues-de-France/La-DGLFLF/Nos-priorites/Rapport-au-Parlement-sur-l-emploi-de-la-langue-francaise-2015, from the Délégation générale à la langue française et aux langues de France of the Ministère de la Culture et de la Communication. In French.
8. DGLFLF: Vers une norme française pour les claviers informatiques. Government Publication (2016), http://www.culture.gouv.fr/Thematiques/Langue-francaise-et-langues-de-France/Politiques-de-la-langue/Langues-et-numerique/Les-technologies-de-la-langue-et-la-normalisation/Vers-une-norme-francaise-pour-les-claviers-informatiques, from the Délégation générale à la langue française et aux langues de France of the Ministère de la Culture et de la Communication. In French.
9. Feit, A.M.: Assignment Problems for Optimizing Text Input. G5 artikkeliväitöskirja (2018), http://urn.fi/URN:ISBN:978-952-60-8016-1
10. Frieze, A., Yadegar, J.: On the quadratic assignment problem. Discrete Applied Mathematics **5**(1), 89 – 98 (1983). https://doi.org/10.1016/0166-218X(83)90018-5
11. Gilmore, P.C.: Optimal and Suboptimal Algorithms for the Quadratic Assignment Problem. SIAM J. Appl. Math. **10**, 305–313 (1962)
12. Gurobi Optimization, L.: Gurobi Optimizer Version 8.1 (2019), http://www.gurobi.com
13. Huber, C., Riedl, W.: The Quadratic Assignment Problem: the Linearization of Xia and Yuan is Weaker than the Linearization of Adams and Johnson and a Family of Cuts to Narrow the Gap, preprint on webpage at https://arxiv.org/abs/1710.02472

14. John, M., Karrenbauer, A.: A Novel SDP Relaxation for the Quadratic Assignment Problem Using Cut Pseudo Bases, pp. 414–425. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-45587-7_36
15. Kaufman, L., Broeckx, F.: An Algorithm for the Quadratic Assignment Problem Using Benders' Decomposition. European Journal of Operational Research **2**(3), 207 – 211 (1978). https://doi.org/10.1016/0377-2217(78)90095-4
16. 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), http://EconPapers.repec.org/RePEc:cwl:cwldpp:4
17. Lawler, E.L.: The Quadratic Assignment Problem. Management Science **9**(4), 586–599 (1963). https://doi.org/10.1287/mnsc.9.4.586
18. Lee, Y., Orlin, J.B.: On Very Large Scale Assignment Problems, pp. 206–244. Springer US, Boston, MA (1994). https://doi.org/10.1007/978-1-4613-3632-7_12
19. Nugent, C., Vollman, T., Ruml, J.: An Experimental Comparison of Techniques for the Assignment of Facilities to Locations. Operations Research **16**(1), 150–173 (1968). https://doi.org/10.1287/opre.16.1.150
20. Peng, J., Mittelmann, H., Li, X.: A new Relaxation Framework for Quadratic Assignment Problems Based on Matrix Splitting. Mathematical Programming Computation **2**(1), 59–77 (2010). https://doi.org/10.1007/s12532-010-0012-6
21. Pollatschek, M., Gershoni, N., Radday, Y.: Optimization of the Typewriter Keyboard by Simulation. Angewandte Mathematik **10** (1976)
22. Povh, J., Rendl, F.: Copositive and Semidefinite Relaxations of the Quadratic Assignment Problem. Discret. Optim. **6**(3), 231–241 (Aug 2009). https://doi.org/10.1016/j.disopt.2009.01.002
23. Queyranne, M.: Performance Ratio of Polynomial Heuristics for Triangle Inequality Quadratic Assignment Problems. Operations Research Letters **4**(5), 231 – 234 (1986). https://doi.org/10.1016/0167-6377(86)90007-6
24. Sherali, H.D., Adams, W.P.: A Hierarchy of Relaxations and Convex Hull Characterizations for Mixed-Integer Zero—One Programming Problems. Discrete Applied Mathematics **52**(1), 83 – 106 (1994). https://doi.org/http://dx.doi.org/10.1016/0166-218X(92)00190-W
25. Xia, Y., Yuan, Y.X.: A new Linearization Method for Quadratic Assignment Problems. Optimization Methods and Software **21**(5), 805–818 (2006). https://doi.org/10.1080/10556780500273077
26. Zhang, H., Beltran-Royo, C., Ma, L.: Solving the Quadratic Assignment Problem by Means of General Purpose Mixed Integer Linear Programming Solvers. Annals OR **207**, 261–278 (2013)
27. Zhao, Q., Karisch, S.E., Rendl, F., Wolkowicz, H.: Semidefinite Programming Relaxations for the Quadratic Assignment Problem. Journal of Combinatorial Optimization **2**(1), 71–109 (1998). https://doi.org/10.1023/A:1009795911987