Noname manuscript No.
(will be inserted by the editor)

On Solving a Hard Quadratic 3-Dimensional Assignment Problem

Hans D. Mittelmann · Domenico Salvagnin

the date of receipt and acceptance should be inserted later

Abstract We address the solution of a very challenging (and previously unsolved) instance of the quadratic 3-dimensional assignment problem, arising in digital wireless communications. The paper describes the techniques developed to solve this instance to optimality, from the choice of an appropriate mixed-integer programming formulation, to cutting planes and symmetry handling. Using these techniques we were able to solve the target instance with moderate computational effort (2.5 million nodes and one week of computations on a standard PC).

Keywords :
Mixed-Integer Programming, Quadratic Assignment, Symmetry

1 Introduction

The axial quadratic 3-dimensional assignment problem (Q3AP) is a generalization of the standard quadratic assignment problem (QAP). In its general form, the problem is defined by a 6-dimensional matrix of costs c_{ijkpqr} of n^6

Hans D. Mittelmann
School of Mathematical and Statistical Sciences, Arizona State University, Tempe, USA
E-mail: mittelmann@asu.edu

Domenico Salvagnin
DEI, University of Padova, Italy
E-mail: salvagni@math.unipd.it

coefficients, and can be formulated (non linearly) as

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{p=1}^{n}\sum_{q=1}^{n}\sum_{r=1}^{n} c_{ijkpqr}x_{ijk}x_{pqr} \tag{1}$$

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

$$\sum_{i=1}^{n}\sum_{k=1}^{n} x_{ijk} = 1 \quad \forall j = 1,\ldots,n \tag{3}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{n} x_{ijk} = 1 \quad \forall i = 1,\ldots,n \tag{4}$$

$$x_{ijk} \in \{0,1\} \tag{5}$$

Equivalently, the objective is to minimize the quadratic objective function (1) over the 3-dimensional assignment polytope defined by (2)−(5). Clearly, the problem is NP-hard, being a generalization of QAP and of the 3-dimensional linear assignment problem (3AP), both of which are NP-hard. Q3AP was introduced almost 50 years ago in a technical memorandum later published in [16], without further development in the literature, and it was recently rediscovered as a tool to model symbol remapping in digital wireless communications [7]. In this context, binary data is mapped to symbols for transmission and, in case of errors, some data packets are retransmitted, following an automatic repeat request (ARQ) protocol. Recent works [20,21] have shown that during retransmission a different encoding should be used, in order to improve the probability of error detection. Intuitively, symbol mapping should be as diversified as possible between retransmissions. When only one retransmission is needed, the problem is equivalent to the solution of a standard QAP. However, if two retransmission are needed, then the problem is a Q3AP (provided that one wants to optimize the simultaneous assignment of binary strings to symbols in all transmissions). In this case, the cost coefficients c_{ijkpqr} represent the costs of assigning to the strings i and p the symbols j and q in the first retransmission and the symbols k and r in the second retransmission. These coefficients can be computed offline based on a probabilistic error model, see [20,21] for details, and can lead to massive improvements in the corresponding communication quality. Note that while a Q3AP of size n has $n! \times n!$ feasible solutions, the optimization has no real time requirements, because the problem needs to be solved in the planning and design phase, and not by the parties involved in the communication. Still, the Q3AP turns out to be an extremely difficult problem to solve: indeed, the exact methods in [7] can solve only instances of size up to 13, even allowing for massive computing power.

The purpose of this paper is to describe our efforts in the solution of the previously unsolved Q3AP real world instance of size 16. In order to successfully solve this instance, we developed a few interesting techniques, that may

turn out to be instrumental for the solution of other Q3AP (or QAP) instances as well. The proposed techniques yield a solution procedure which is exact in nature: we decompose the original instance into subproblems based on symmetry arguments (which are symbolic, and thus exact, in nature), and then we formulate each subproblem as mixed-integer programming (MIP) program, to be solved with a complete method such as branch-and-cut. However, since each subproblem is solved by a state-of-the-art, but finite precision, MIP solver, the implemented procedure can only guarantee an optimality proof up to a given tolerance. In the following, by optimality proof we will refer to this computational connotation, highlighting when needed all the precautions that we took in order to increase the numerical precision and fault tolerance of our procedure.

The outline of the paper is as follows: Section 2 describes the real world instance that we wanted to solve, while sections 3 to 5 present the techniques that we developed for the purpose, namely a lightweight mixed-integer programming formulation (Section 3), symmetry handling (Section 4) and cutting plane generation (Section 5). Section 6 reports the computational experiments. Finally, conclusions are drawn in Section 7.

2 Instance

The most common digital modulation techniques are

- phase-shift keying (PSK) using a finite number of phases
- frequency-shift keying (FSK) using a finite number of frequencies
- amplitude-shift keying (ASK) using a finite number of amplitudes
- quadrature amplitude modulation (QAM) using at least two phases and at least two amplitudes

In the case of PSK, ASK or QAM, where the carrier frequency of the modulated signal is constant, the modulation alphabet is often conveniently represented in a constellation diagram. For an n symbol PSK or n-PSK that would show n points equidistantly with respect to angle distributed on the unit circle. This is a simple pattern and one that leads to a high degree of symmetry. For QAM along the x-axis would be the inphase signal, for example a cosine waveform, and along the y-axis the quadrature phase signal, for example a sine waveform, both amplitude modulated.

The constellation points of the QAM modulation are thus better distributed, the distance between them is a factor of 1.62 in case of n=16 larger than that of 16-PSK. This leads to 16-QAM requiring a 4.19dB lower signal-to-noise ratio than 16-PSK. Consequently 8-PSK is commonly used in the IEEE 802 standards for wireless communications and also n-QAM with n=16 and higher. An interesting challenge would thus also to solve to optimality the Q3APs associated with the QAM methods. Here, however, we are attracted by the high symmetry of the PSK method, which allows to apply and further

refine symmetry exploiting methods that have lately been developed for general mixed-integer programs. The data for this real world instance is available in [6].

Regarding exact solution of the Q3APs associated with ARQ modulation techniques, so far only the 8-PSK case has been solved. For n=16 cases only the successive optimization of the symbol mappings, instead of the simultaneous one as done here, it leads to several standard QAPs, has been considered in [21]. Originally the cost matrix for the 16-PSK case is real. As is customary, its elements were scaled by 10^{16} and rounded to integers: this introduces an approximation error that is orders of magnitude smaller than the default tolerances used by any commercial MIP solver, so it is absolutely safe in our context. From now on, we will refer to this all-integer version as our instance to solve. Table 1 lists some of its characteristics. While we show in the present paper that it is possible to solve the Q3AP derived from 16-PSK, bigger instances such as 32-QAM, 64-QAM, etc can only be treated with a reasonable effort by heuristic methods to obtain feasible solutions, combined with lower-bounding techniques to show which gap in objective value is present. For 2D index assignment problems up to dimension 512 this was done in [23], based on methods developed in [13,15]. The approach can be generalized to Q3AP problems.

Table 1 Instance characteristics.

n	16
n^3	4,096
n^6	16,777,216
non-zero objective coefficients	12,755,712
objective sparsity	76.03%
objective dynamism	$3.627 \cdot 10^{12}$
symmetry group order	49,152
non-zero scaled objective coefficients	7,524,096
scaled objective sparsity	44.85%
scaled objective dynamism	$3.627 \cdot 10^{6}$

3 MIP Model

In order to solve such a large Q3AP instance with MIP technology, a lightweight, yet sufficiently strong, formulation is needed. A trivial way to linearize the quadratic terms in the objective function is to introduce additional binary variables $\psi_{ijkpqr} = x_{ijk}x_{pqr}$, and the corresponding linking constraints. However, in our case this amounts to introducing $\Theta(n^6)$ variables and constraints, which is hopeless for $n = 16$. To overcome this issue, we preferred to extend the lightweight QAP Kaufman and Broeckx (KB) model [9] to the Q3AP case.

The model requires only n^3 artificial continuous variables, defined as

$$w_{ijk} = \left(\sum_{p=1}^{n} \sum_{q=1}^{n} \sum_{r=1}^{n} c_{ijkpqr} x_{pqr} \right) x_{ijk}$$

which can be easily linearized with big-M coefficients. The corresponding MIP model reads

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} w_{ijk} \tag{6}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ijk} = 1 \quad \forall k \in \{1, \ldots, n\} \tag{7}$$

$$\sum_{i=1}^{n} \sum_{k=1}^{n} x_{ijk} = 1 \quad \forall j \in \{1, \ldots, n\} \tag{8}$$

$$\sum_{j=1}^{n} \sum_{k=1}^{n} x_{ijk} = 1 \quad \forall i \in \{1, \ldots, n\} \tag{9}$$

$$w_{ijk} \geq \sum_{p=1}^{n} \sum_{q=1}^{n} \sum_{r=1}^{n} c_{ijkpqr} x_{pqr} - M(1 - x_{ijk}) \quad \forall (i,j,k) \in \{1, \ldots, n\}^3 \tag{10}$$

$$x_{ijk} \in \{0, 1\} \tag{11}$$

$$w_{ijk} \geq 0 \tag{12}$$

Note that if all cost coefficients are integer, as in our case, articifial variables w_{ijk} could be declared integer as well, yielding a pure integer formulation. This model is known (in its QAP version) to be of little use in practice because of the big-M constraints (10). In particular, it can be proved [24] that the root-node bound is always zero. However, a much stronger formulation can be obtained by adding to (6)−(12) the (polynomial) family of cutting planes

$$w_{ijk} \geq L_{ijk} x_{ijk} \tag{13}$$

where each L_{ijk} is defined as the optimal value of the 3AP

$$\min \sum_{p=1}^{n} \sum_{q=1}^{n} \sum_{r=1}^{n} c_{ijkpqr} x_{pqr} \tag{14}$$

$$\sum_{p=1}^{n} \sum_{q=1}^{n} x_{pqr} = 1 \quad \forall r \in \{1, \ldots, n\} \tag{15}$$

$$\sum_{p=1}^{n} \sum_{r=1}^{n} x_{pqr} = 1 \quad \forall q \in \{1, \ldots, n\} \tag{16}$$

$$\sum_{q=1}^{n} \sum_{r=1}^{n} x_{pqr} = 1 \quad \forall p \in \{1, \ldots, n\} \tag{17}$$

$$x_{ijk} = 1 \tag{18}$$

$$x_{pqr} \in \{0, 1\} \tag{19}$$

Note that, contrary to the QAP case [3,24], in order to compute the coefficients of this family of cutting planes, we need to solve a 3AP for each cut, which is by itself an NP-hard problem. Still, for the instance at hand, those 3AP instances were straightforward to solve with state-of-the-art MIP technology. In addition, the symmetry of the instance can be exploited to solve much less than n^3 3APs (see Section 4). The KB model, together with the family of cutting planes (13), was used recently in [3] to solve highly symmetric QAP instances, for the same reasons. Finally, we note that the big-M coefficients in constraints (10) can also be computed by solving 3APs akin to the model just described, with the only differences that maximization is used instead of minimization, and the fixing $x_{ijk} = 1$ is replaced with $x_{ijk} = 0$. While these 3APs turn out to be no harder to solve in practice than those needed to compute coefficients L_{ijk}, and the same symmetry trick applies, their computation is not even needed if constraints (10) are modeled as indicator constraints, a feature implemented in most commercial MIP solvers.

4 Symmetry Handling

We applied a standard symmetry detection algorithm (such as, e.g., Nauty [12]) to the model (6)−(12), and found a non trivial symmetry group of order 49,152. While not an extreme case of symmetry by any measure, this group is still big enough to slow down the search significantly, in particular given that each branch-and-cut node is quite expensive in our case. According to the symmetry group, the set of binary variables x_{ijk} can be partitioned into 6 orbits (the same applies to the artificial variables w), whose main figures are reported in Table 2. As anticipated in the previous section, we can exploit this knowledge to significantly reduce the effort needed to compute coefficients L_{ijk} and the big-Ms in the model, given that they will clearly be the same for all the variables in the same orbit. This reduces the number of 3APs to solve from 8,192

down to 12. Note that there is no chicken-and-egg problem here, because the symmetry group of the model can be computed *before* knowing the values of the coefficients (cuts (13) are not yet in the model, and all big-Ms can be set to the same value).

Table 2 Orbit characteristics.

Orbit	Size	L	bigM
0	256	0	6.4325e+06
1	1,024	2	6.8216e+06
2	512	1,199	6.8349e+06
3	1,024	30	9.2371e+06
4	1,024	3,544	9.6247e+06
5	256	46,465	1.2042e+07

Symmetry has long been recognized as a curse for the traditional enumeration approaches used in many optimization communities—we refer to [11, 5] for recent surveys on the subject. Various techniques for dealing with symmetric problems have been studied, with *isomorphism pruning* [10] and *orbital branching* [14] being the most popular and effective strategies deployed within MIP solvers. In a preliminary computational study, we implemented both symmetry breaking techniques: however, their effect was still not sufficient to solve our instance to optimality.

A different approach to symmetry management has been proposed recently and is called *orbital shrinking* [2]: the main idea is to aggregate variables along orbits, in order to obtain a smaller (and symmetry free) reformulation. While this approach can yield an exact reformulation in some cases, in general it provides only a relaxation of the original problem: however, orbital shrinking has been turned into a complete method in [18]. The algorithm in [18] is a decomposition scheme, where a *master* problem is used to enumerate the feasible solutions of the orbital shrinking reformulation of the model, and a *slave* problem is solved for each such solution to check if it can indeed be turned into a solution of the original problem (see [18] for details). Orbital shrinking has been shown to be a successful approach in solving symmetric optimization problems with a medium amount of symmetry [18, 19]: intuitively, for the method to be effective, the shrunken reformulation must be sufficiently easy to solve, yet retain enough of the structure of the original model. As such, it is a natural candidate for our case, where the amount of symmetry is not extreme.

The exact method in [18] can, in principle, be applied to pure-integer as well as mixed-integer problems: however, in the mixed-integer case, one should enumerate, in the master, all possible values also for the continuous variables, which makes the method impractical. Even in our case, where as noted we could declare variables w_{ijk} as integer, the number of solutions to check would be enormous. Thus, in order to apply orbital shrinking, we had to modify

the method in the following way. Given the orbital shrinking reformulation of the model (which is obtained by aggregating variables and constraints along orbits, and, by definition, has one variable for each variable orbit and one constraint for each constraint orbit), we removed the aggregated constraints corresponding to (10), the aggregated continuous variables, and the objective function. The final model (which by construction is still a relaxation) reads:

$$y_0 + y_1 + y_2 + y_3 + y_4 + y_5 = 16 \tag{20}$$
$$y_1 + 2y_3 + y_4 = 16 \tag{21}$$
$$y_2 + y_4 + 2y_5 = 8 \tag{22}$$
$$2y_0 + y_1 + y_2 = 8 \tag{23}$$
$$y_i \in \{0, \dots, |O_i| - 1\} \quad \forall i \in \{0, \dots, 5\} \tag{24}$$

where we denoted with y_i the sum of all the binary variables x_{ijk} in orbit O_i, i.e.

$$y_i = \sum_{(i,j,k) \in O_i} x_{ijk}$$

Model (20)-(24) is of no use as far as the bound is concerned; however, it can be used to enumerate all possible 6-tuples of y variables. Each 6-tuple $\overline{y}^t$ can then be used to construct a subproblem in which the sums of the variables along the orbits are fixed to $\overline{y}^t$. It is easy to see that this is a partition of the feasible space of the original problem. Luckily enough, in our case there are only 85 different 6-tuples (reported in Table 3), which implies that we need to solve *only* 85 subproblems.

It is important to note that each subproblem, by construction, has the same symmetry group as the original problem, a property shared with the exact algorithm in [18]. Usually, it is possible to get around this issue by exploiting ad-hoc symmetry breaking procedures for the subproblem at hand: however, in our case, we just resorted to traditional general purpose techniques such as orbital branching and isomorphism pruning. To sum up, the proposed strategy exploits the symmetry group of the original model twice:

1. computation of the orbital shrinking inspired model (20)-(24)
2. solution of each subproblem with isomorphism pruning

Finally, we note that the constraints used to fix the values of the orbits in each subproblem can be taken into account when computing the coefficients L_{ijk}, resulting in an additional strengthening of the formulation.

5 Cutting Planes

Cutting planes (13) turn out to be crucial for the strength of the KB formulation at the root node. Still, the resulting formulation is not strong enough for a pure branch-and-bound to succeed. If we dig deeper into the structure

of this family of cutting planes, we can clearly see the weak spot: each L_{ijk} is obtained by solving, independently, a linear 3AP problem, thus the corresponding assignments are not *synchronized* among the n^3 subproblems. Indeed, if all assignments were the same, then the KB model would give the value of the optimal solution: unfortunately, this is, by far, not case. A possible way to (partially) fix this issue is to keep on separating local versions of cuts (13) throughout the search tree, using the current domain of the variables for the 3APs. This is beneficial because, at a given node of the tree, all the assignments must at least agree on the variables that are fixed at that node. Of course, this also means that all cutting planes generated this way are only locally valid, so some bookkeeping is needed in order to guarantee correctness. The same strategy was also exploited in [3], where, however, each cut could be separated much faster by an ad-hoc polynomial procedure, much in the spirit of the classical Gilmore-Lawler bound computations [1], while in our case each 3AP is solved by a black box MIP solver.

While computational experiments show that separating local cuts of the family (13) is indeed very beneficial for the resulting enumerative search, still this is not enough to solve our instance. This is easily explained by the fact that we need to dive deep into the tree for the local domains to significantly constrain the separation problems. Intuitively, fixing a few variables does not yield sufficiently strong cuts, and the dual bound is still too weak to prune nodes early on.

Thus, we devised a new family of cutting planes, which is a generalization of (13). While a given cutting plane of the form (13) provides a lower bound on the value of a variable w_{ijk}, given the current domains of variables x and assuming that $x_{ijk} = 1$, the new family gives lower bounds on the sums of the form $w_{ijk} + w_{pqr}$, assuming that $x_{ijk} = 1$ and $x_{pqr} = 1$. The resulting cuts are all of the form:

$$w_{ijk} + w_{pqr} \geq T_{ijkpqr}(x_{ijk} + x_{pqr} - 1) \tag{25}$$

and the coefficient T_{ijkpqr} can be obtained as the optimal value of the following MIP:

$$\min \quad w_{ijk} + w_{pqr} \tag{26}$$

$$\sum_{s=1}^{n}\sum_{t=1}^{n} x_{pqr} = 1 \quad \forall u \in \{1,\dots,n\} \tag{27}$$

$$\sum_{s=1}^{n}\sum_{u=1}^{n} x_{pqr} = 1 \quad \forall t \in \{1,\dots,n\} \tag{28}$$

$$\sum_{t=1}^{n}\sum_{u=1}^{n} x_{pqr} = 1 \quad \forall s \in \{1,\dots,n\} \tag{29}$$

$$w_{ijk} \geq \sum_{s=1}^{n}\sum_{t=1}^{n}\sum_{u=1}^{n} c_{ijkstu} x_{stu} \tag{30}$$

$$w_{pqr} \geq \sum_{s=1}^{n}\sum_{t=1}^{n}\sum_{u=1}^{n} c_{pqrstu} x_{stu} \tag{31}$$

$$x_{ijk} = 1 \tag{32}$$

$$x_{pqr} = 1 \tag{33}$$

$$x_{stu} \in \{0,1\} \tag{34}$$

Note that by construction $T_{ijkpqr} \geq L_{ijk} + L_{pqr}$. In our computational experience, each cut of the family (25) is not significantly harder to separate than a cut (13), yet care must be taken because the number of possible cuts at each node is n^6, which, although polynomial, could make an exact separation too slow in practice. Finally, note that, as in the previous section, the constraints used to fix the values of the orbits in each subproblem can be taken into account when computing the coefficients of cuts (13) and (25), both globally and locally.

6 Computations

A lot of tuning was needed to perfect the techniques presented in the previous sections, that we will describe in the next subsections.

6.1 Primal Heuristics

After the first runs on the complete model (6)−(12), we realized that even a state-of-the-art solver such as CPLEX has lots of trouble in finding a good quality (hopefully optimal) feasible solution early on in the search. This is of course affecting negatively performance in general, and the more so in our case where the LP relaxation is quite weak. In addition, this is a huge issue for the decomposition scheme: not knowing the globally optimal solution may

turn some easy subproblems into very difficult ones, and carrying over the best solution from one subproblem to the next is not an option (we would not be able to process subproblems in parallel, and in any case the sequential order would be arbitrary).

In order to overcome this issue, we implemented an ad-hoc primal heuristic for the Q3AP, along the lines of the ILS method described in [7,22]. Surprisingly, even a very simple implementation was able to consistently find the (later proven) optimal solution of value $207,462,238,240$ in a matter of minutes. As such, we can, for practical purposes, assume that we know the optimal solution in advance, and concentrate on the task of proving its optimality. For reference, we provide the optimal solution below:

$$
\begin{array}{l}
0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15 \\
0\ 1\ 2\ 3\ 4\ 6\ 5\ 7\ 8\ 10\ 9\ 11\ 12\ 13\ 14\ 15 \\
6\ 14\ 2\ 10\ 8\ 12\ 0\ 4\ 11\ 15\ 3\ 7\ 5\ 13\ 1\ 9
\end{array}
$$

6.2 MIP Model

A first issue is given by model (6)−(12) itself: because the objective 6-dimensional matrix is quite dense, the corresponding constraints (10) are dense as well. In addition, the dynamism of these coefficients is so high (values range from 1 to $3.627 \cdot 10^{12}$) that the LP relaxation is not only very slow to solve but also numerically unstable. Thus, we decided to scale (and round down) all objective coefficients by a factor of 10^6. This proved to be quite beneficial, with the objective density going from 76% to 45% and the final dynamism being $3.627 \cdot 10^6$. Still, rounding may change the set of optimal solutions to our problem, in particular if we demand strict tolerances (for example, the optimal objective of $207,462,238,240$ is mapped to $207,392$ and not to $207,462$). In order to maintain the correctness of the method regardless of scaling, we implemented the following strategy:

- each subproblem is solved with an objective cutoff of $208,000$.
- feasible solutions are collected but not reported as feasible to the underlying MIP solver, thus forcing the enumeration of all feasible solutions of (scaled) value less than $208,000$, barring symmetry.
- all collected solutions are evaluated with the original objective function and the best one is marked as the optimal solution.

Finally, constraints (10) are not added to the initial formulation, but implemented as indicator constraints, resulting in orders of magnitude savings in the LP solution times.

6.3 Symmetry Breaking

A preliminary comparison between orbital shrinking and isomorphism pruning showed that the latter is more effective in pruning the search tree, provided

that the right branching strategy is used. After some tuning, we found out that a good strategy is to rank the variables x_{ijk} by decreasing values of L_{ijk}, using the variable index to break ties. The rationale behind this choice is that in this way we improve the bound quite quickly, because the variables with higher priority are the more expensive ones, but at the same time those whose contribution is more underestimated by the linear relaxation.

6.4 Cutting Planes

We separate local versions of both cutting planes (13) and (25) throughout the tree. However, care must be taken in order to limit the separation overhead, which can be quite substantial (even orders of magnitude in node throughput). We implemented the following strategy:

- the separation procedure for cuts (13) is called at the current node only if the number of variables fixed to 1 is between 2 and 12, and only for those x variables that are fractional in the LP relaxation
- cuts (13) are added only if a sufficient number (say 10) of them is violated at the current node. This is to spare the bookkeeping overhead needed by local cuts when only a few can be added. We consider a cut violated if the new coefficient L_{ijk} is greater than $1.1w^*_{ijk}/x^*_{ijk}$, i.e., it must improve upon the current (implied) coefficient by at least 10%.
- the separation procedure for cuts (25) is called at the current node only if the number of variables fixed to 1 is between 2 and 12 and even. In addition, the first variable x_{ijk} must be fixed to 1, while the second variable x_{pqr} must be fractional in the LP relaxation. Notice that, given this assumption, we can simplify the cut as

$$w_{ijk} + w_{pqr} \geq T_{ijkpqr} x_{pqr}$$

- also cuts (25) are added only if a sufficient number of them is violated at the current node. We consider a cut violated if the new coefficient T_{ijkpqr} is such that

$$T_{ijkpqr} x^*_{pqr} > 1.1(w^*_{ijk} + w^*_{pqr})$$

- only the LP relaxations of the sub-MIPs needed to compute the coefficients of cuts (13) and (25) are solved, without resorting to enumeration. This has several benefits: (i) an occasionally challenging sub-MIP will not slow down the overall procedure and (ii) the coefficients are weaker and thus the chances of generating a (slightly) invalid cut because of numerics and tolerances are lower.

6.5 Results

We implemented our codes in C++, using IBM ILOG CPLEX 12.5.1 [8] as black box MIP solver through the CPLEX callable library APIs and Permlib [17]

to implement the group operations needed by isomorphism pruning. All tests have been performed on a cluster of 24 identical machines, each equipped with an Intel Xeon E3-1220v2 running at 3.10GHz and with 16GB of RAM.

Computing the symmetry group, as well as enumerating the 85 solutions of model (20)-(24), was done in less than one minute (using the constraint programming solver Gecode [4]), and thus can be considered negligible. All sub-MIPs (to be more precise, their LP relaxations) are always solved with default CPLEX tolerances. As far as the subproblem resolution is concerned, we used again default tolerances and disabled all CPLEX cut separators, in order to speedup node throughtput (CPLEX cuts were in any case not effective because of symmetry) and improve numerical safety. Detailed results are reported in Table 3.

Table 3: Subproblems characteristics.

Subproblem	Time (s)	Nodes
0,0,8,8,0,0	52,545	599,725
0,1,7,7,1,0	54,943	351,345
0,2,6,6,2,0	45,635	227,453
0,2,6,7,0,1	6,813	22,543
0,3,5,5,3,0	22,055	118,927
0,3,5,6,1,1	13,970	35,419
0,4,4,4,4,0	24,246	133,046
0,4,4,5,2,1	4,150	15,256
0,4,4,6,0,2	89	163
0,5,3,3,5,0	24,977	104,091
0,5,3,4,3,1	1,471	9,025
0,5,3,5,1,2	88	15
0,6,2,2,6,0	18,261	72,035
0,6,2,3,4,1	758	3,099
0,6,2,4,2,2	76	15
0,6,2,5,0,3	20	0
0,7,1,1,7,0	7,398	35,643
0,7,1,2,5,1	245	569
0,7,1,3,3,2	78	15
0,7,1,4,1,3	45	0
0,8,0,0,8,0	452	3,944
0,8,0,1,6,1	103	41
0,8,0,2,4,2	62	15
0,8,0,3,2,3	31	0
0,8,0,4,0,4	14	0
1,0,6,7,2,0	1,843	10,676
1,0,6,8,0,1	818	6,593
1,1,5,6,3,0	6,430	30,187

Table 3 continued

Subproblem	Time (s)	Nodes
1,1,5,7,1,1	3,258	12,047
1,2,4,5,4,0	20,217	108,980
1,2,4,6,2,1	1,688	5,940
1,2,4,7,0,2	102	81
1,3,3,4,5,0	33,871	121,652
1,3,3,5,3,1	1,398	6,070
1,3,3,6,1,2	116	17
1,4,2,3,6,0	27,664	87,038
1,4,2,4,4,1	775	2,697
1,4,2,5,2,2	99	15
1,4,2,6,0,3	24	0
1,5,1,2,7,0	12,675	46,127
1,5,1,3,5,1	266	433
1,5,1,4,3,2	107	21
1,5,1,5,1,3	47	0
1,6,0,1,8,0	2,464	10,909
1,6,0,2,6,1	154	55
1,6,0,3,4,2	87	19
1,6,0,4,2,3	33	0
1,6,0,5,0,4	17	0
2,0,4,6,4,0	2,494	14,946
2,0,4,7,2,1	226	623
2,0,4,8,0,2	32	25
2,1,3,5,5,0	16,951	92,512
2,1,3,6,3,1	614	1,653
2,1,3,7,1,2	109	15
2,2,2,4,6,0	20,930	94,663
2,2,2,5,4,1	502	1,493
2,2,2,6,2,2	99	15
2,2,2,7,0,3	23	0
2,3,1,3,7,0	11,955	45,794
2,3,1,4,5,1	238	333
2,3,1,5,3,2	101	15
2,3,1,6,1,3	46	0
2,4,0,2,8,0	2,748	10,899
2,4,0,3,6,1	114	43
2,4,0,4,4,2	91	15
2,4,0,5,2,3	34	0
2,4,0,6,0,4	17	0
3,0,2,5,6,0	1,753	10,368
3,0,2,6,4,1	78	49
3,0,2,7,2,2	56	13
3,0,2,8,0,3	11	0

Table 3 continued

Subproblem	Time (s)	Nodes
3,1,1,4,7,0	3,627	16,384
3,1,1,5,5,1	111	39
3,1,1,6,3,2	84	13
3,1,1,7,1,3	48	0
3,2,0,3,8,0	1,152	4,732
3,2,0,4,6,1	83	31
3,2,0,5,4,2	69	13
3,2,0,6,2,3	34	0
3,2,0,7,0,4	17	0
4,0,0,4,8,0	101	177
4,0,0,5,6,1	43	15
4,0,0,6,4,2	21	0
4,0,0,7,2,3	21	0
4,0,0,8,0,4	6	0
Total	457,312	2,476,819

According to the table, all subproblems required a total of $457,312$ seconds (less than a week) if performed on a serial machine with the same speed. The total number of nodes turned out to be just a little below 2.5 million, surprisingly low given the difficulty of the instance, proving that the techniques developed in the paper were very successful in pruning the enumeration tree. We note that the results refer to the case in which each CPLEX run was given 4 threads and the search mode was set to CPX_PARALLEL_OPPORTUNISTIC, resulting in a nondeterministic run. This choice is motivated by the fact that because of our expensive separation procedures, the default load balancing policy of CPLEX resulted in a severe underuse of the available cores. Indeed, forcing deterministic search results in a very similar outcome as far as the number of nodes is concerned, but with a doubled running time.

For reference, we report that a default CPLEX run on model (6)−(12), without all the techniques described here, but with the global cutting planes (13) added at the very beginning and the optimal solution provided as incumbent, yields a dual bound of only $11,934$ (a depressing relative gap of 94.25% left to close) after 3 days of computations, more than 35 millions of enumerated nodes and 120GB of memory occupied.

7 Conclusions

We addressed the solution of a very challenging (and previously unsolved) instance of the quadratic 3-dimensional assignment problem, arising in digital wireless communications. In order to solve this instance to optimality, we developed novel techniques, such as a new family of cutting planes and a

two-level symmetry handling strategy based on orbital shrinking. These techniques, among others, proved to be instrumental in the successful solution of this Q3AP instance. More importantly, they are general purpose in nature and can be easily extended to different Q3AP or even QAP instances with similar properties.

Acknowledgements We thank Peter M. Hahn for providing the numerical data for the 16-PSK Q3AP instance. The work of the first author was supported in part by AFOSR under grant FA9550-12-1-0153.

References

1. Burkard, R., Dell'Amico, M., Martello, S.: Assignment Problems. SIAM (2009)
2. Fischetti, M., Liberti, L.: Orbital shrinking. In: A.R. Mahjoub, V. Markakis, I. Milis, V.T. Paschos (eds.) ISCO, *Lecture Notes in Computer Science*, vol. 7422, pp. 48–58. Springer (2012)
3. Fischetti, M., Monaci, M., Salvagnin, D.: Three ideas for the quadratic assignment problem. Operations Research **60**(4), 954–964 (2012)
4. Gecode Team: Gecode: Generic constraint development environment (2012). Available at `http://www.gecode.org`
5. Gent, I.P., Petrie, K.E., Puget, J.F.: Symmetry in constraint programming. In: F. Rossi, P. van Beek, T. Walsh (eds.) Handbook of Constraint Programming, pp. 329–376. Elsevier (2006)
6. Hahn, P.M.: URL `http://www.seas.upenn.edu/~hahn/`
7. Hahn, P.M., Kim, B.J., Stützle, T., Kanthak, S., Hightower, W.L., Samra, H., Ding, Z., Guignard, M.: The quadratic three-dimensional assignment problem: Exact and approximate solution methods. European Journal of Operational Research **184**(2), 416–428 (2008)
8. IBM: IBM ILOG Cplex Optimization Studio. `http://www.cplex.com`
9. Kaufman, L., Broeckx, F.: An algorithm for the quadratic assignment problem using Benders' decomposition. European Journal of Operational Research **2**, 204–211 (1978)
10. Margot, F.: Exploiting orbits in symmetric ILP. Mathematical Programming **98**(1), 3–21 (2003)
11. Margot, F.: Symmetry in integer linear programming. In: M. Jünger, T. Liebling, D. Naddef, G. Nemhauser, W. Pulleyblank, G. Reinelt, G. Rinaldi, L. Wolsey (eds.) 50 Years of Integer Programming 1958-2008, pp. 647–686. Springer Berlin Heidelberg (2010)
12. McKay, B.D.: Practical graph isomorphism (1981)
13. Mittelmann, H.D., Peng, J.: Estimating bounds for quadratic assignment problems associated with hamming and manhattan distance matrices based on semidefinite programming. SIAM Journal on Optimization **20**(6), 3408–3426 (2010)
14. Ostrowski, J., Linderoth, J., Rossi, F., Smriglio, S.: Orbital branching. Mathematical Programming **126**(1), 147–178 (2011)
15. Peng, J., Mittelmann, H.D., Li, X.: A new relaxation framework for quadratic assignment problems based on matrix splitting. Mathematical Programming Computation **2**(1), 59–77 (2010)
16. Pierskalla, W.P.: The multi-dimensional assignment problem. Operations Research **16**(2), 422–431 (1968)
17. Rehn, T.: Fundamental permutation group algorithms for symmetry computation. Master's thesis, Otto-von-Guericke University Magdeburg (2010)
18. Salvagnin, D.: Orbital shrinking: a new tool for hybrid MIP/CP methods. In: CPAIOR, pp. 204–215 (2013)
19. Salvagnin, D., Walsh, T.: A hybrid MIP/CP approach for multi-activity shift scheduling. In: CP, pp. 633–646 (2012)

20. Samra, H., Ding, Z.: Symbol mapping diversity in iterative decoding/demodulation of ARQ systems. In: ICC, pp. 3585–3589. IEEE (2003)
21. Samra, H., Ding, Z., Hahn, P.M.: Symbol mapping diversity design for multiple packet transmissions. IEEE Transactions on Communications **53**(5), 810–817 (2005)
22. Stützle, T.: Iterated local search for the quadratic assignment problem. European Journal of Operational Research **174**(3), 1519–1539 (2006)
23. Wu, X., Mittelmann, H.D., Wang, X., Wang, J.: On computation of performance bounds of optimal index assignment. IEEE Transactions on Communications **59**(12), 3229–3233 (2011)
24. Xia, Y., Yuan, Y.: A new linearization method for quadratic assignment problem. Optimization Methods and Software **21**, 803–816 (2006)