A dual framework for lower bounds of the quadratic assignment problem based on linearization*

Stefan E. Karisch[†] Eranda Çela[‡] Jens Clausen[†] Torben Espersen[§]

March 3, 1998

Abstract

A dual framework allowing the comparison of various bounds for the quadratic assignment problem (QAP) based on linearization, e.g. the bounds of Adams and Johnson, Carraresi and Malucelli, and Hahn and Grant, is presented. We discuss the differences of these bounds and propose a new and more general bounding procedure based on the dual of the linearization of Adams and Johnson. The new procedure has been applied to problems of dimension up to $n = 72$, and the computational results indicate that the new bound competes well with existing linearization bounds and yields a good trade off between computation time and bound quality.

Key words. Combinatorial optimization, quadratic assignment problem, lower bounds, dual approach.

AMS subject classifications. 90C27, 90B80, 90C11.

1 Introduction

The Quadratic Assignment Problem (QAP) is in its simplest form concerned with locating facilities on locations such that total transportation costs are minimized. The transportation cost incurred by locating two facilities is proportional both to the flow of transportation between the facilities and the distance between their locations. Even this simple form, called the Koopmans-Beckmann form of the problem, is strongly NP-hard and recognized to be among the most difficult NP-hard problems to solve to optimality.

In the more general form of QAP [16], the cost coefficients are not products of a flow and a distance term. Furthermore, also cost coefficients modeling fixed cost incurred by locating a facility on a specific location are present. Denoting the cost incurred by locating facility i on location j *and* facility k on location l by d_{ijkl}, and the fixed cost of locating facility i on location j by c_{ij}, an integer programming formulation of the problem is :

*This manuscript is available as SFB Report 120, Department of Mathematics, Technical University Graz, Austria and as Technical Report IMM-REP-1998-02, Department of Mathematical Modeling, Technical University of Denmark.

†Technical University of Denmark, Department of Mathematical Modeling, Building 321, DK-2800 Lyngby, Denmark

‡Technical University Graz, Department of Mathematics, Steyrergasse 30, A-8010 Graz, Austria.

§University of Copenhagen, Department of Computer Science, Universitetsparken 1, DK-2100 Copenhagen, Denmark

$$
\begin{array}{lll}
\text{minimize} & \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \sum_{l=1}^{n} d_{ijkl} x_{ij} x_{kl} + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} & \\
\text{subject to} & \sum_{j=1}^{n} x_{ij} = 1 \qquad 1 \leq i \leq n & \\
& \sum_{i=1}^{n} x_{ij} = 1 \qquad 1 \leq j \leq n & \\
& x_{ij} \in \{0,1\} \qquad 1 \leq i, j \leq n. &
\end{array} \tag{1}
$$

The integer programming formulation for Koopmans-Beckmann QAPs is obtained by defining d_{ijkl} to be $a_{ik}b_{jl}$, where a_{ik} is the flow from facility i to facility k and b_{jl} is the distance from location j to location l.

In addition to the integer programming formulation, QAP is often formulated combinatorially in terms of permutations. Formulated this way the combinatorial structure of the set of feasible solutions becomes more clear, however solution techniques from integer and linear programming are no longer immediately applicable. Defining $\mathcal{S}_n$ to be the set of permutations of $\{1, 2, \ldots, n\}$, the combinatorial formulation is:

$$
\min_{\phi \in \mathcal{S}_n} \sum_{i=1}^{n} \sum_{k=1}^{n} d_{i\phi(i)k\phi(k)} + \sum_{i=1}^{n} c_{i\phi(i)}. \tag{2}
$$

Also Koopmans-Beckmann QAPs can be formulated combinatorially:

$$
\min_{\phi \in \mathcal{S}_n} \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik} b_{\phi(i)\phi(k)} + \sum_{i=1}^{n} c_{i\phi(i)}. \tag{3}
$$

Again, the interpretation of the model becomes more clear – if $j = \phi(i)$ and $l = \phi(k)$ then facility i is located on location j, facility k on location l and the cost incurred is $a_{ik}b_{jl}$.

We assume without loss of generality that d_{ijkl} are nonnegative. If they are not, we can add a sufficiently large constant to all d_{ijkl} which does not change the optimal permutation and increases the objective function only by n^2 times the added constant. Furthermore, we can assume without loss of generality that the costs d_{ijij} equal 0 for all $1 \leq i, j \leq n$. If this is not the case, then we can modify the coefficients of the given QAP by setting $c_{ij} := c_{ij} + d_{ijij}$ and $d_{ijij} := 0$ for all $1 \leq i, j \leq n$, and the QAP with modified coefficients would clearly be equivalent to the original one.

QAP problems are normally solved using Branch-and-Bound, an exhaustive search method, which dynamically generates the search tree and searches as large parts of this as possible only implicitly through the use of bounds. A bound function is a function, which for a given QAP returns a lower bound for the value of the optimal solution to the problem. There is a trade-off between quality and computational ease for bound functions in that better bounds are more time consuming to compute. Therefore, the choice of bound function is crucial for the quality of the resulting Branch-and-Bound algorithm. If the bound is too weak, the search tree grows too large, and if the bound is very tight, it may not be computable in the available time.

Over the years, a large number of bound functions have been proposed. Many of these are based on the idea of LP-relaxation – to reformulate QAP in such a way that LP-techniques can be used in computing the bound. The interrelationship between these bounds have been commented upon by a number of authors, but due to the lack of a common framework for the description of the individual bounds, the relation between the bounds has been unclear, both with respect to methodology and bound quality. In the current paper we derive such a framework. This enables us to describe and understand the similarities and differences between the considered bounding procedures, and to propose new bounding procedures generalizing these.

The paper is organized as follows. In Section 2 we review bounds based on linearization, namely the Gilmore-Lawler bound [12, 16], the bounds of Carraresi and Malucelli [5] and Assad and Xu [3], the bounds of Adams and Johnson [1], and the bounds of Hahn and Grant [13, 14]. Section 3 presents a dual framework which does not only allow the comparison of the different bounds but which also motivates new bounding techniques. A new bound is presented and computationally tested in Section 4 and conclusions are drawn in Section 5. We include an appendix, which contains small sized examples for the individual bounds.

2 Bounds based on linearization

There is a large number of bounding techniques for the QAP which are based on equivalent formulations of the problem as a mixed integer linear program (MILP). The basic idea is to solve the continuous relaxation of the MILP to optimality or to solve the dual of this relaxation approximately by using standard techniques such as Lagrangian methods or steepest ascent procedures. For more details and a general overview on different MILP formulation for the QAP and bounds based on them the reader is referred to Pardalos, Rendl and Wolkowicz [19] and Çela [6].

In this section we shortly review the most popular lower bounds for the QAP related to such ideas: the Gilmore-Lawler bound [12, 16], the bound of Carraresi and Malucelli [5], the bound of Assad and Xu [3], the bounds of Adams and Johnson [1], and the bounds of Hahn and Grant [13]. As will become clear in Section 3, all these bounds can be obtained by approximately solving an MILP formulation for the QAP which is denoted by LP:

$$
\text{LP}\qquad
\begin{array}{lll}
\text{minimize} & \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}\sum\limits_{\substack{k=1\\ j\neq i}}^{n}\sum\limits_{\substack{l=1\\ l\neq j}}^{n} d_{ijkl}y_{ijkl} + \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} c_{ij}x_{ij} & \\
\text{subject to} & \sum\limits_{\substack{l=1\\ l\neq j}}^{n} y_{ijkl} = x_{ij} \qquad 1 \le i,j,k \le n, i \neq k & (4)\\
& \sum\limits_{\substack{k=1\\ k\neq i}}^{n} y_{ijkl} = x_{ij} \qquad 1 \le i,j,l \le n, j \neq l & (5)\\
& \sum\limits_{j=1}^{n} x_{ij} = 1 \qquad 1 \le i \le n & (6)\\
& \sum\limits_{i=1}^{n} x_{ij} = 1 \qquad 1 \le j \le n & (7)\\
& y_{klij} = y_{ijkl} \qquad 1 \le i,j,k,l \le n, i < k, l \neq j & (8)\\
& x_{ij} \in \{0,1\} \qquad 1 \le ij \le n & \\
& y_{ijkl} \ge 0 \qquad 1 \le i,j,k,l \le n, k \neq i, l \neq j &
\end{array}
$$

LP was proposed by Adams and Johnson [1] as a modification of an other well known MILP formulation for the QAP, the MILP formulation of Frieze and Yadegar [11]. The linearization is obtained by introducing new variables $y_{ijkl} = x_{ij}x_{kl}$ and premultiplying the assignment constraints by x_{kl}. By showing that for each feasible solution (x, y) of LP, we have $y_{ijkl} = x_{ij}x_{kl}$, for $1 \leq i, j, k, l \leq n$, it can be proven that LP is equivalent to the QAP as given in (1). For a formal proof of this fact the reader is referred to e.g. Adams and Sherali [2]. We are going to describe the structure of LP in more detail in Section 2.1.

Adams and Johnson [1] argue that all the above mentioned bounding techniques (except the bound of Hahn and Grant [13] which was proposed later) are directly related to LP as they approximately solve the dual of the continuous relaxation of LP. The continuous relaxation of LP which is obtained by replacing the Boolean constraints on the variables x_{ij} by nonnegativity constraints, will play a special role throughout the rest of this paper, and will be denoted by CLP.

The recently proposed bounds of [13] turned out to be competitive with the best previously existing bounding techniques both in terms of the bounds' quality and running time requirements. At a first glance this bounding technique seems to be purely combinatorial and not related to LP. However, it is a dual approach to approximately solve the continuous relaxation of LP and fits in the general dual framework to be described in Section 3.

2.1 The structure of LP

By construction, the new variables in LP correspond to $y_{ijkl} = x_{ij}x_{kl}$. In other words, the four dimensional array $Y = (y_{ijkl})$ can be thought as being an $n^2 \times n^2$ matrix, representing the Kronecker product of two permutation matrices $X \otimes X$. This interpretation allows for partitioning Y into n^2 blocks of $n \times n$ matrices $Y^{(i,j)} = x_{ij}X$. For $n = 3$, the array Y presented as a 9×9 matrix looks as follows.

$$Y = \left(\begin{array}{ccc|ccc|ccc}
y_{1111} & y_{1112} & y_{1113} & y_{1211} & y_{1212} & y_{1213} & y_{1311} & y_{1312} & y_{1313} \\
y_{1121} & y_{1122} & y_{1123} & y_{1221} & y_{1222} & y_{1223} & y_{1321} & y_{1322} & y_{1323} \\
y_{1131} & y_{1132} & y_{1133} & y_{1231} & y_{1232} & y_{1233} & y_{1331} & y_{1332} & y_{1323} \\
\hline
y_{2111} & y_{2112} & y_{2113} & y_{2211} & y_{2212} & y_{2213} & y_{2311} & y_{2312} & y_{2313} \\
y_{2121} & y_{2122} & y_{2123} & y_{2221} & y_{2222} & y_{2223} & y_{2321} & y_{2322} & y_{2323} \\
y_{2131} & y_{2132} & y_{2133} & y_{2231} & y_{2232} & y_{2233} & y_{2331} & y_{2332} & y_{2333} \\
\hline
y_{3111} & y_{3112} & y_{3113} & y_{3211} & y_{3212} & y_{3213} & y_{3311} & y_{3312} & y_{3313} \\
y_{3121} & y_{3122} & y_{3123} & y_{3221} & y_{3222} & y_{3223} & y_{3321} & y_{3322} & y_{3323} \\
y_{3131} & y_{3132} & y_{3133} & y_{3231} & y_{3232} & y_{3233} & y_{3331} & y_{3332} & y_{3333}
\end{array} \right)$$

Since Y inherits properties of the permutation matrices, one can exploit their structure in LP. A $\{0,1\}$-matrix $X = (x_{ij})$ which satisfies the constraints of (1) is called a permutation matrix. To each permutation ϕ of $\{1, 2, \ldots, n\}$ corresponds a permutation matrix $X_\phi = (x_{ij}^\phi)$ where $x_{ij}^\phi = 1$ if and only if $\phi(i) = j$. Vice-versa, for each permutation matrix X there is a permutation ϕ_X which maps index i to some j ($\phi_X(i) = j$) if and only if $x_{ij} = 1$.

Firstly, elements of the form $y_{ijil} = x_{ij}x_{il}$, with $j \neq l$, or $y_{ijkj} = x_{ij}x_{kj}$, with $i \neq k$ are always zero, because there is exactly one nonzero in each row and column of a permutation matrix, and therefore they do not contribute to the objective function. These elements are "disallowed" and not considered as variables in LP. From the construction of Y and the fact that permutation matrices are binary, we have $y_{ijij} = x_{ij}x_{ij} = x_{ij}$. We call element x_{ij} (or equivalently y_{ijij}) the *leader element* of block $Y^{(i,j)}$. This observation allows us to assume that $d_{ijij} = 0$ for all (i, j) since we would otherwise add it to c_{ij}. We call the c_{ij} the *leader costs*. Furthermore, we have $y_{ijkl} = x_{ij}x_{kl} = x_{kl}x_{ij} = y_{klij}$, implying the so called *complementarity* *constraints* (8) in LP. y_{ijkl} and y_{klij} are called *complementary elements*, while d_{ijkl} and d_{klij} are *complementary costs.*

If we replace disallowed entries by "$*$" and print leader elements x_{ij} in bold face, the array Y for $n = 3$ looks as follows:

$$Y = \left(\begin{array}{ccc|ccc|ccc} \mathbf{x_{11}} & * & * & * & \mathbf{x_{12}} & * & * & * & \mathbf{x_{13}} \\ * & y_{1122} & y_{1123} & y_{1221} & * & y_{1223} & y_{1321} & y_{1322} & * \\ * & y_{1132} & y_{1133} & y_{1231} & * & y_{1233} & y_{1331} & y_{1332} & * \\ \hline * & y_{2112} & y_{2113} & y_{2211} & * & y_{2213} & y_{2311} & y_{2312} & * \\ \mathbf{x_{21}} & * & * & * & \mathbf{x_{22}} & * & * & * & \mathbf{x_{23}} \\ * & y_{2132} & y_{2133} & y_{2231} & * & y_{2233} & y_{2331} & y_{2332} & * \\ \hline * & y_{3112} & y_{3113} & y_{3211} & * & y_{3213} & y_{3311} & y_{3312} & * \\ * & y_{3122} & y_{3123} & y_{3221} & * & y_{3223} & y_{3321} & y_{3322} & * \\ \mathbf{x_{31}} & * & * & * & \mathbf{x_{32}} & * & * & * & \mathbf{x_{33}} \end{array}\right)$$

Taking a closer look at the constraints of LP, it is easy to see that the assignment constraints for X, (6) and (7), correspond to assignment constraints on the leader elements. We define the so called *leader LAP*

$$l\text{-LAP} \qquad \begin{array}{lll} \text{minimize} & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} & \\ \text{subject to} & \sum_{j=1}^{n} x_{ij} = 1 & 1 \le i \le n \\ & \sum_{i=1}^{n} x_{ij} = 1 & 1 \le j \le n \\ & x_{ij} \ge 0 & 1 \le i, j \le n. \end{array} \qquad (9)$$

The constraints within the $(n-1) \times (n-1)$ dimensional submatrix of $Y^{(i,j)}$, containing only allowed elements, read

$$\begin{array}{ll} \sum_{\substack{l=1 \\ l \ne j}}^{n} y_{ijkl} = x_{ij} & 1 \le k \le n, k \ne i \\ \sum_{\substack{k=1 \\ k \ne i}}^{n} y_{ijkl} = x_{ij} & 1 \le l \le n, l \ne j \end{array}$$

and correspond to (4) and (5). If we fixed $x_{ij} = 1$ for some pair (i, j), these become just assignment constraints on $Y^{(i,j)}$ and hence motivate the definition of an $(n-1)$ dimensional *block LAP*, denoted by $\text{LAP}^{(i,j)}$, and given as

$$\text{LAP}^{(i,j)} \qquad \begin{array}{lll} \text{minimize} & \sum_{\substack{k=1 \\ k \ne i}}^{n} \sum_{\substack{l=1 \\ l \ne j}}^{n} d_{ijkl} y_{ijkl} & \\ \text{subject to} & \sum_{\substack{l=1 \\ l \ne j}}^{n} y_{ijkl} = 1 & 1 \le k \le n, k \ne i \\ & \sum_{\substack{k=1 \\ k \ne i}}^{n} y_{ijkl} = 1 & 1 \le l \le n, l \ne j \\ & y_{ijkl} \ge 0 & 1 \le k, l \le n. \end{array} \qquad (10)$$

In the remainder of the paper we do not give the range of the indices i, j, k, and l explicitly but assume that they go from 1 to n.

2.2 The Gilmore-Lawler bound

The Gilmore-Lawler bound, shortly denoted by GLB, is one of the most popular bounds for the QAP. It was proposed by Gilmore [12] for the Koopmans-Beckmann QAP (3), and by Lawler [16] for the more general QAP as given in (1). For the general QAP, GLB can be computed by solving (n^2+1) LAPs. Let z_{ij}, for some (i,j), be the optimal value of LAP$^{(i,j)}$ with cost coefficients d_{ijkl}. Then GLB is obtained as the optimal value of a leader LAP with cost coefficients given as $c_{ij}+z_{ij}$, i.e. by

$$\begin{array}{ll} \text{minimize} & \sum_i \sum_j (c_{ij}+z_{ij})x_{ij} \\ \text{subject to} & \sum_j x_{ij} = 1 \qquad \forall i \\ & \sum_i x_{ij} = 1 \qquad \forall j \\ & x_{ij} \geq 0 \qquad \forall (i,j). \end{array} \tag{11}$$

Since the computation of GLB involves solving (n^2+1) LAPs, the time complexity amounts to $O(n^5)$. It is well known that this time complexity can be reduced to $O(n^3)$ in the case of Koopmans-Beckmann QAPs (see eg. [12]).

Even though GLB is usually derived as a pure combinatorial bound of the QAP, it can also be obtained in terms of CLP. It is not difficult to see and it has been proven by Adams and Johnson [1] that if the complementarity constraints (8) are dropped from CLP, the remaining relaxed problem can be solved by decomposing it into (n^2+1) LAPs in the way described above. The optimal value of this relaxed version of CLP is exactly GLB.

2.3 The bounds of Carraresi and Malucelli and Assad and Xu

Another bounding procedure for the QAP which can be obtained in terms of the linearization LP has been proposed by Carraresi and Malucelli [5]. This procedure, which with a special parameter setting produces also the bound of Assad and Xu [3], relies on so-called *reformulations* of the QAP. A reformulation of a QAP in the form (1) is another QAP with coefficients d'_{ijkl} and c'_{ik}, where

$$d'_{ijkl} = d_{ijkl} + \beta_{ijkl} - \sigma_{kli} - \theta_{klj} + \gamma_{kl} \tag{12}$$

$$c'_{ij} = c_{ij} + \sum_{k\neq i} \sigma_{ijk} + \sum_{l \neq j} \theta_{ijl} - (n-1)\gamma_{ij} \tag{13}$$

for all (i,j,k,l), and for each permutation ϕ of $\{1,2,\ldots,n\}$ we have

$$\sum_i \sum_{k\neq i} d_{i\phi(i)k\phi(k)} + \sum_i c_{i\phi(i)} = \sum_i \sum_{k\neq i} d'_{i\phi(i)k\phi(k)} + \sum_i c'_{i\phi(i)}.$$

The idea is to reformulate iteratively the given QAP and compute the Gilmore-Lawler bound for each reformulation. According to the definition of a reformulation, each of these GLBs will be a lower bound to the original problem and we can choose the best out of them. By specifying the initial values β^0_{ijkl}, σ^0_{ijk}, θ^0_{ijl} and γ^0_{ij} for the parameters and the formulas to update these parameters in each iteration one obtains different bounds for the QAP. Assad and Xu [3] proposed one of these bounds, shortly denoted by AXB, where

$$\beta^0_{ijkl} = 0,\ \sigma^0_{ijk} = 0,\ \theta^0_{ijl} = 0,\ \gamma^0_{ij} = 0,$$

for all (i,j,k,l). The parameters β_{ijkl}, σ_{ijk} and θ_{ijl} are kept equal to zero during all iterations, whereas the parameters γ_{ij}^{s+1} at iteration $(s+1)$ are obtained from the parameters γ_{ij}^{s} at iteration s as follows:

$$\gamma_{ij}^{s+1} = \gamma_{ij}^{s} + \frac{1}{n-1} z_{ij}(\gamma^s),$$

where for all pairs (i,j), $z_{ij}(\gamma^s)$ is the optimal solution of a block LAP with costs $(d_{ijkl} - \gamma_{ij}^s)$, i.e.

$$\mathrm{LAP}_{\gamma}^{(i,j)} \qquad \begin{array}{lll} \text{minimize} & \sum\limits_{k\neq i}\sum\limits_{l\neq j}(d_{ijkl} - \gamma_{ij}^s) y_{ijkl} & \\ \text{subject to} & \sum\limits_{l\neq j} y_{ijkl} = 1 & \forall k, k\neq i \\ & \sum\limits_{k\neq i} y_{ijkl} = 1 & \forall l, l\neq j \\ & y_{ijkl} \geq 0 & \forall (k,l), k\neq i, l\neq j. \end{array} \tag{14}$$

Other bounds based on reformulations were proposed by Carraresi and Malucelli [5]. The most popular one, shortly denoted by CMB, uses the optimal values of the dual variables of LAPs as given in (14) in its update formula. Similarly as in the case of AXB, Carraresi and Malucelli start with values equal to 0 for all parameters in the first iteration. In the following iterations σ_{ijk}^{s+1} and θ_{ijl}^{s+1} are set equal to the optimal values of the dual variables of $\mathrm{LAP}_{\gamma}^{(i,j)}$ for each pair of indices (i,j). The update formulas for the other parameters are as follows:

$$\beta_{ijkl}^{s+1} = d_{klij} - d_{ijkl} \quad \forall (i,j,k,l)$$

$$\gamma_{ij}^{s+1} = \frac{1}{n-1}(c_{ij} - \lambda_i - \mu_j) \quad \forall (i,j),$$

where λ_i, μ_j are the optimal values of the dual variables of the leader LAP given below

$$\begin{array}{lll} \text{minimize} & \sum\limits_{i}\sum\limits_{j}\left[c_{ij} + \sum\limits_{k\neq i}^{n}\sigma_{ijk}^s + \sum\limits_{l\neq j}^{n}\theta_{ijl}^s - (n-1)\gamma_{ij}^s\right] x_{ij} & \\ \text{subject to} & \sum\limits_{j} x_{ij} = 1 & \forall i \\ & \sum\limits_{i} x_{ij} = 1 & \forall j \\ & x_{ij} \geq 0 & \forall (i,j). \end{array} \tag{15}$$

Obviously, GLB is obtained by just applying the first iteration of bounding procedures based on reformulations (be that the bound of Assad and Xu or the bound of Carraresi and Malucelli), where the initial values of all parameters β_{ijkl}, σ_{ijk}, θ_{ijl} and γ_{ij} are equal to 0.

The time complexity of such bounding procedures amounts to $O(n^5)$ per iteration as one GLB computation for a QAP of the form (1) is the dominating part in each iteration.

2.4 The bounds of Adams and Johnson

Adams and Johnson [1] consider the continuous relaxation of LP denoted by CLP. CLP has $2n^2(n-1) + n^2(n-1)^2/2 + 2n$ constraints, in addition to the nonnegativity constraints for all variables, and $n^2[(n-1)^2+1]$ variables. Clearly, the optimal value of CLP is a lower bound for LP and equivalently for QAP. However, due to the huge number of variables and constraints, it is practically impossible to solve CLP to optimality even for QAPs of a moderate size. Instead of solving CLP to optimality, Adams and Johnson apply a steepest ascent procedure to compute "good" feasible solutions of a Lagrangian dual of CLP. The value of the objective function corresponding to such solutions is of course a lower bound for the optimal value of CLP, and hence, also for the optimal value of the QAP.

The Lagrangian dual is obtained by relaxing the complementarity constraints (8) which are multiplied by Lagrange multipliers α_{ijkl}, for all (i,j,k,l) with $i<k$ and $j \neq l$, and are added to the objective function. This Lagrangian dual is denoted by CLP_α and is given as

$$\mathrm{CLP}_\alpha \qquad \begin{array}{lll} \text{minimize} & \sum_i \sum_j \sum_{k>i} \sum_{l \neq j} (d_{ijkl} - \alpha_{ijkl}) y_{ijkl} & \\ & \qquad + \sum_i \sum_j \sum_{k<i} \sum_{l \neq j} (d_{ijkl} + \alpha_{klij}) y_{ijkl} + \sum_i \sum_j c_{ij} x_{ij} & \\ \text{subject to} & \sum_{l \neq j} y_{ijkl} = x_{ij} & \forall (i,j,k), k \neq i \\ & \sum_{k \neq i} y_{ijkl} = x_{ij} & \forall (i,j,l), l \neq j \\ & \sum_{j=1}^{n} x_{ij} = 1 & \forall (i,j) \\ & \sum_{i=1}^{n} x_{ij} = 1 & \forall (i,j) \\ & x_{ij} \geq 0 & \forall (i,j) \\ & y_{ijkl} \geq 0 & \forall (i,j,k,l), k \neq i, l \neq j \end{array}$$

For any fixed set of Lagrange multipliers α, solving the relaxation CLP_α of CLP is equivalent to computing GLB with costs as given in the objective function, i.e. the solution is obtained by solving a leader LAP with cost coefficients $c_{ij} + z_{ij}$, here denoted by l-LAP_α, where z_{ij} is the optimal value of the block LAP given by

$$\mathrm{LAP}_\alpha^{(i,j)} \qquad \begin{array}{lll} \text{minimize} & \sum_{k>i} \sum_{l \neq k} (d_{ijkl} - \alpha_{ijkl}) y_{ijkl} + \sum_{k<i} \sum_{l \neq k} (d_{ijkl} + \alpha_{klij}) y_{ijkl} & \\ \text{subject to} & \sum_{l \neq j} y_{ijkl} = 1 & \forall k, k \neq i \\ & \sum_{k \neq i} y_{ijkl} = 1 & \forall l, l \neq j \\ & y_{ijkl} \geq 0 & \forall (k,l), k \neq i,\ l \neq j, \end{array}$$

for each pair (i,j).

The dual ascent procedure of Adams and Johnson, abbreviated by DAP, consists then of iteratively solving CLP_α, where the Lagrange multipliers are updated in each iteration based on sensitivity analysis arguments. A bound is obtained after each iteration of the ascent procedure. In the first iteration all Lagrange multipliers are set to 0. The update formula involves the dual variables β_{ijkl} associated with the nonnegativity constraints in $\mathrm{LAP}_\alpha^{(i,j)}$, and the dual variables γ_{ij} corresponding to the nonnegativity constraints $x_{ij} \geq 0$ of l-LAP_α. The formula involves also a set of control parameters φ_{ijkl} and ψ_{ij} and is given by

$$\alpha_{ijkl}^{s+1} = \alpha_{ijkl}^{s} + (1 - \varphi_{ijkl})[\beta_{ijkl}(\alpha^s) + \frac{\psi_{ij}}{n-1}\gamma_{ij}(\alpha^s)] - \varphi_{ijkl}[\beta_{klij}(\alpha^s) + \frac{\psi_{kl}}{n-1}\gamma_{kl}(\alpha^s)] \qquad (16)$$

for all (i,j,k,l), with $i<k$ and $l \neq j$. $\beta_{ijkl}(\alpha^s)$, $\gamma_{ij}(\alpha^s)$ are the optimal values of the dual variables obtained for Lagrange multipliers equal to α^s. It is easily seen that after running the first iteration of the DAP with Lagrange multipliers equal to 0 we obtain GLB.

In an effort to improve the performance of DAP, Adams and Johnson modify it to obtain the so called *enhanced dual ascent procedure* (EDAP). Here, the dual multipliers are not only changed after each iteration but also *within* each iteration, where an iteration consists of solving the n^2, $(n-1)$-dimensional problems $\mathrm{LAP}_{\alpha}^{(i,j)}$, and then l-LAP_{α}. In [1], the block problems $\mathrm{LAP}_{\alpha}^{(i,j)}$ are solved in nondecreasing lexicographic order of the pairs (i, j), i.e. $\mathrm{LAP}_{\alpha}^{(1,1)}$, $\mathrm{LAP}_{\alpha}^{(1,2)}$, ..., $\mathrm{LAP}_{\alpha}^{(1,n)}$, $\mathrm{LAP}_{\alpha}^{(2,1)}$, ..., $\mathrm{LAP}_{\alpha}^{(n,n)}$. For each pair (i, j) after solving $\mathrm{LAP}_{\alpha}^{(i,j)}$, the α-s are updated by setting $\alpha_{ijkl} := \alpha_{ijkl} + \beta_{ijkl}$ and $\beta_{ijkl} := 0$, for all (k, l) such that $k > i$ and $l \neq j$. This update affects only α-s involved in the current block LAP and block LAPs to be solved *next*. To do the modification of β_{ijkl}, the optimal solution of $\mathrm{LAP}_{\alpha}^{(i,j)}$ remains an optimal solution of this problems also after the update. As for the other block LAPs affected by the modification they are solved at a later stage. Thus, similarly as for DAP, after each iteration of EDAP we have a feasible solution of the Lagrangian dual.

In principle any fixed prespecified order of processing the block LAPs could be chosen in EDAP, but we would have to be careful with the update of the α-s within the iterations, so as to change only those α-s which affect only block LAPs to be solved *next* besides the current block LAP.

After solving $\mathrm{LAP}_{\alpha}^{(i,j)}$ to obtain $\gamma_{ij}(\alpha^s)$ and $\beta_{ijkl}(\alpha^s)$, and if $i < k$, modify α^s_{ijkl} by setting $\alpha^s_{ijkl} = \alpha^s_{ijkl} + \beta_{ijkl}(\alpha^s)$, and set the corresponding multiplier $\beta_{ijkl} = 0$. Adams and Johnson show that EDAP produces a lower bound to CLP and hence to the original QAP. Both DAP and EDAP solve $(n^2 + 1)$ LAPs per iteration, and therefore, the time complexity amounts to $O(n^5)$ per iteration.

In [1], Adams and Johnson use a specific setting for the control parameters φ_{ijkl} and ψ_{ij}, namely $\varphi_{ijkl} = \frac{1}{2}$ and $\psi_{ij} = \frac{n-1}{n}$. We denote the bounds that result for this setting by AJB when using DAP and AJB' when applying EDAP.

As for the relationship between CMB and AJB, it has been pointed out in [1] that the procedure of Carraresi and Malucelli produces a feasible solution to the dual of CLP. As such, CMB can not be better than the optimal value of CLP, as mentioned also by Carraresi and Malucelli in [5]. However, it is not straightforward to see that CMB arises as a feasible solution of the dual of CLP, and Adams and Johnson do not really argue upon this point. Moreover, Adams and Johnson do not solve CLP to optimality, but rather give a lower bound to this problem by computing a presumably good feasible solution of its Lagrangian dual. In the next section we show that both bounding procedures can be interpreted as heuristic algorithms for solving the dual of CLP in a common, and therefore a more general framework.

2.5 The bound of Hahn and Grant

Recently, a new bounding procedure which combines GLB ideas with reduction steps (see [6] for more information and references on so-called reduction methods) has been proposed by Hahn and Grant [13, 14]. Consider a QAP of the form (2). The four dimensional array $D = (d_{ijkl})$ can also be thought as being an $n^2 \times n^2$ matrix composed of n^2 submatrices $D^{(i,j)}$. Using the structure of the Kronecker product of permutation matrices again, one can remove elements of D which are associated with disallowed elements of Y. Recall that we have assumed w.l.o.g. that $d_{ijij} = 0$. Moreover, since the costs c_{ij} contribute to the objective function if and only if $x_{ij} = y_{ijij} = 1$, we replace d_{ijij} by c_{ij}. To give an example, for $n = 3$ the array D with entries of C in the places of the leader elements looks as follows:

$$D = \left(\begin{array}{ccc|ccc|ccc}
\mathbf{c_{11}} & * & * & * & \mathbf{c_{12}} & * & * & * & \mathbf{c_{13}} \\
* & d_{1122} & d_{1123} & d_{1221} & * & d_{1223} & d_{1321} & d_{1322} & * \\
* & d_{1132} & d_{1133} & d_{1231} & * & d_{1233} & d_{1331} & d_{1332} & * \\
\hline
* & d_{2112} & d_{2113} & d_{2211} & * & d_{2213} & d_{2311} & d_{2312} & * \\
\mathbf{c_{21}} & * & * & * & \mathbf{c_{22}} & * & * & * & \mathbf{c_{23}} \\
* & d_{2132} & d_{2133} & d_{2231} & * & d_{2233} & d_{2331} & d_{2332} & * \\
\hline
* & d_{3112} & d_{3113} & d_{3211} & * & d_{3213} & d_{3311} & d_{3312} & * \\
* & d_{3122} & d_{3123} & d_{3221} & * & d_{3223} & d_{3321} & d_{3322} & * \\
\mathbf{c_{31}} & * & * & * & \mathbf{c_{32}} & * & * & * & \mathbf{c_{33}}
\end{array}\right)$$

The bounding procedure acts on D and uses the following classes of operations:

(R1) Add a constant to all allowed entries of some row (column) of some submatrix $D^{(i,j)}$ and either subtract the same constant from the allowed entries of another row (column) of the same submatrix, or subtract it from the leader in that submatrix.

(R2) Add a constant to all allowed entries of some row (column) of the $n^2 \times n^2$ matrix D.

The procedure iteratively applies reduction operations of class R1 or R2 to the cost array of the problem at hand. The key observation is that the operations of class R1 maintain the values of the objective function unchanged, but redistribute the entries of the submatrices $D^{(i,j)}$, whereas operations of class R2 add some constant to the objective function value corresponding to each permutation. Hence, operations of class R2 maintain the ordering of the permutations with respect to the value of the objective function. The main idea of the bounding procedure is then the following. If the operations applied to D decrease the objective function value corresponding to each permutation by an amount r and are performed in such a way that the entries of the transformed array D' remain nonnegative, then, clearly, r is a lower bound for the optimal solution of the given QAP. If, moreover, the 0-entries in the transformed matrix D' comply with the pattern of zeros in the Kronecker product $X_\phi \otimes X_\phi$ for some permutation matrix X_ϕ, then r is the optimal objective function value of the original QAP and permutation ϕ is an optimal solution to QAP (2).

The procedure developed to find such a lower bound r, or possibly, to optimally solve the problem, is essentially similar to the Hungarian method for the linear assignment problem. It uses operations of classes R1 and R2 to redistribute the entries of D so as to obtain a pattern of zeros which complies with the pattern of zeros of the Kronecker product $X_\phi \otimes X_\phi$ for some permutation matrix X_ϕ. It is an iterative approach which starts by applying the Hungarian method to $(n-1)\times(n-1)$ submatrices of the matrices $D^{(i,j)}$, obtained by canceling the leader together with its corresponding row and column, i.e. the method solves block LAPs in the submatrices.

The most characteristic feature of the procedure proposed by Hahn and Grant which also seems to be the main reason for its good performance is the "interaction" between complementary costs. Recall, that for each 4-tuple of indices (i,j,k,l), the elements d_{ijkl} and d_{klij} are complementary due to the complementarity constraints in LP. The interaction is performed in the following way. Before applying the Hungarian method to the current block LAP, all entries d_{ijkl} of the corresponding cost matrix are updated by adding to them the corresponding complementary elements (be this original d_{ijkl}-s or already updated ones.) After having solved all n^2 block LAPs (obtained after respective complementary costs interactions), the cost matrices of the blocks contain the reduced costs, and the optimal values of LAP$^{(i,j)}$ are added to the corresponding leaders. Then, the Hungarian method is applied to the matrix of the leaders to solve the leader LAP. The matrix of the leaders is then transformed by the Hungarian method, i.e. it contains the reduced cost of the leader LAP. The optimal value r of the leader LAP is added to the bound b, which is initially set to 0.

As mentioned above, the current b is a lower bound for the original problem. Further, if the pattern of zeroes of the transformed array D, that is the array resulting after the above (n^2+1) applications of the Hungarian method, coincides with the pattern of zeros of $X_\phi \otimes X_\phi$, for some permutation matrix X_ϕ, the procedure stops and outputs b as the optimal value of the considered QAP. If this is not the case, the algorithm checks whether the (transformed) matrix of leaders contains some non-zero elements. If it does not, the procedure terminates with a lower bound b for the original problem. Otherwise, the reduced costs corresponding to non-zero leaders, i.e. the current non-zero entries of the matrix of the leaders, are distributed uniformly over the elements of the corresponding submatrices by applying operations of class R1. After this redistribution the current iteration is over and the next iteration starts with the application of the Hungarian method to each of the (transformed) matrices $D^{(i,j)}$ after having collected the complementary elements. The whole process is repeated until the value of the bound b can not be improved any more or some other termination criterion is fulfilled. In summary, the whole process is a repeated computation of Gilmore-Lawler bounds on iteratively transformed problem data. The time complexity of each iteration is basically that of the GLB for a QAP of the form (1), i.e. $O(n^5)$. The bound produced by this procedure is shortly denoted by HGB.

A closer look at the procedure of Hahn and Grant reveals that – due to the interaction between complementary elements – the value of the lower bound depends on the order in which the block LAPs are solved. In the first iteration the block LAPs are processed in lexicographic order as for AJB'. In any other iteration s, $s > 1$, Hahn and Grant first process the block LAPs whose corresponding (transformed) leaders have been equal to 0 at the end of the previous iteration in lexicographic order. Then the remaining block LAPs are also processed in lexicographic order.

An other idea proposed in [13] concerns the interaction between a pair of complementary elements. As reported in the paper, especially good results are obtained if approximately 50% of the sum of the complementary costs go to the element whose block LAP is processed, while the remaining portions gets to the complementary element. Finally, there is one more idea corresponding the *redistribution of parts of the bound*. After each iteration a part of the bound is redistributed uniformly to the entries of the $n^2 \times n^2$ matrix D (row-wise or column-wise). As the number of iterations increases the part of the bound being redistributed becomes becomes smaller and smaller. Hahn and Grant experiment with an exponential decrease of the part of the bound which is distributed to the entries of D. The bound obtained by the procedure of Hahn and Grant *with redistribution of parts of the bound* and only partial transfer between complementary costs is denoted by HGB' throughout the rest of the paper.

Although at a first glance HGB and HGB' are not derived by approximately solving the relaxation CLP, we show in the next section that each of these bounds is equal to the value of the objective function of some feasible solution to the dual of CLP.

3 Bounds based on a dual framework to solve CLP

In this section we will introduce a dual framework to approximate the optimal value of CLP. We will show that all the bounds described in the previous section fit into this framework and are yielded by feasible solutions of the dual of CLP denoted by DCLP. DCLP is given as follows:

$$
\text{DCLP}\qquad
\begin{array}{lll}
\text{maximize} & \sum\limits_{i=1}^{n} \lambda_i + \sum\limits_{j=1}^{n} \mu_j & \\
\text{subject to} & \sigma_{ijk} + \theta_{ijl} + \alpha_{ijkl} + \beta_{ijkl} = d_{ijkl} & \forall (i,j,k,l), i<k, l \neq j \\
 & \sigma_{ijk} + \theta_{ijl} - \alpha_{klij} + \beta_{ijkl} = d_{ijkl} & \forall (i,j,k,l), i>k, l \neq j \\
 & \lambda_i + \mu_j - \sum\limits_{\substack{k=1\\k\neq i}}^{n} \sigma_{ijk} - \sum\limits_{\substack{l=1\\l\neq j}}^{n} \theta_{ijl} + \gamma_{ij} = c_{ij} & \forall (i,j) \\
 & \beta_{ijkl} \geq 0 & \forall (i,j,k,l), k \neq i, l \neq j \\
 & \gamma_{ij} \geq 0 & \forall (i,j).
\end{array}
$$

For a fixed pair (i,j), σ_{ijk} and θ_{ijl} are the dual variables for the equalities (4) and (5), respectively. Using the structural terms introduced in the previous section for LP, σ_{ijk} and θ_{ijl} would correspond to dual variables for the block LAPs. The dual variables for the equalities (6) and (7) are λ_i and μ_j, respectively, which do correspond to the dual variables of the leader LAP. α_{ijkl} are the dual variables for the complementarity conditions (8), and β_{ijkl} and γ_{ij} are the slacks, i.e. the dual variables to the nonnegativity constraints for y_{ijkl} and x_{ij}. Within the structural framework of LP, β_{ijkl} are the slacks for the dual of the block LAP in the block (i,j), whereas γ_{ij} are the slacks for the dual of the LAP of the leaders.

From weak duality, it follows, that every feasible solution of DCLP provides a lower bound for the quadratic assignment problem. In Section 3.1 we show that GLB, CMB, AJB, AJB' and HGB can be derived as values of the objective function of DCLP corresponding to some feasible solutions of it. We present Algorithm 3.1 which produces the existing bounds in terms of feasible solutions of DCLP. Then in Section 3.2 we derive a general solution procedure for DCLP and show that all considered bounds can be obtained by different settings of some free parameters in this general procedure. Since the enhanced bound of Hahn and Grant [13], denoted by HGB', is more involved than the other ones, it is not described in the frame of Algorithm 3.1, but only within the more general framework to be presented in Section 3.2. In Section 3.3 we compare our general bounding procedure presented in Section 3.2 with the bounding schemes of Adams and Johnson, reviewing the relationship between these schemes, GLB and CMB as already discussed in [1].

3.1 A dual framework for existing bounds

Each of the bounds GLB, CMB, AJB, AJB', HGB, HGB' can be obtained as a feasible solution of DCLP produced by some iterative algorithm to solve DCLP. All these algorithms are quite similar from a conceptual point of view. We describe them within a common solution procedure and specify in each step what each of the algorithms does. This helps to understand the differences and the similarities between the considered bounds. In turn, this allows us to give a more general procedure to solve DCLP which includes as special cases all algorithms mentioned above. This general procedure is presented in the next section.

The iterative algorithms for DCLP which are described in the following have several properties in common. First, they all start with a feasible solution for DCLP and iteratively improve this solution by performing a prespecified number of iterations (parameter max_it in the algorithm below). Secondly, in each iteration the feasible solution for the dual of DCLP is found in a Gilmore-Lawler fashion: for each pair of indices (i, j) the dual problem of $\mathrm{LAP}^{(i,j)}$ is solved. Then a leader LAP with "transformed and updated" costs c_{ij} is solved to obtain a feasible solution of DCLP, and hence, the current lower bound. Moreover, in the case of CMB, AJB', HGB and HGB', the complementary costs d_{ijkl} and d_{klij} (see Section 2.5) are treated together at some stage of the algorithm. Again, the idea is to exploit the fact that either both of these coefficients contribute to the objective function (in the case that $y_{ijkl} = 1$), or none of them does ($y_{ijkl} = 0$).

In order to simplify the presentation, we split up the variables θ_{ijl}, σ_{ijk}, λ_i and μ_j into a "fixed" and a "variable" part, i.e.

$$\begin{aligned}
&\sigma_{ikl} \to \sigma'_{ijk} + \sigma_{ijk}\\
&\theta_{ijl} \to \theta'_{ijl} + \theta_{ijl}\\
&\lambda_i \to \lambda'_i + \lambda_i\\
&\mu_j \to \mu'_j + \mu_j.
\end{aligned}$$

The variables σ'_{ijk}, θ'_{ijl}, λ'_i, μ'_j are used to move costs from the σ_{ijk}, θ_{ijl}, λ_i, μ_j to the corresponding slacks (β_{ijkl} or γ_{ij}) in order to update the current dual solution. Besides these variables we have the parameters φ_{ijkl} and ψ_{ij}, $1 \le i, j, k, l \le n$, which serve to model the update of the variables α_{ijkl} in the case of AJB and AJB'.

Algorithm 3.1 (Algorithm for bounds except HGB')

Step 0 ALL BOUNDS: Input the size of the problem n, costs D and C, and the maximal number of iterations max_it. Initialize $\alpha_{ijkl} = \theta_{ijl} = \theta'_{ijl} = \sigma_{ikl} = \sigma'_{ikl} = \lambda_i = \lambda'_i = \mu_j = \mu'_j = 0$, $\beta_{ijkl} = d_{ijkl}$, and $\gamma_{ij} = c_{ij}$. Set iteration counter $s = 1$.
GLB: Set max_it = 1.
AJB & AJB': Input φ_{ijkl}, ψ_{ij}.

Step 1 ALL BOUNDS but HGB: For all pairs (i, j) in lexicographic order do the following substeps of Step 1.

HGB: For all pairs (i, j) in some prespecified order do the following substeps of Step 1.

Step 1a HGB: For all (k, l), $k \ne i$, $l \ne j$, merge complementary costs via

$$\begin{aligned}
\alpha_{ijkl} &= \alpha_{ijkl} - \beta_{klij}, \quad \beta_{klij} = 0, \quad \text{if } i < k\\
\alpha_{klij} &= \alpha_{klij} + \beta_{klij}, \quad \beta_{klij} = 0, \quad \text{if } i > k.
\end{aligned}$$

Step 1b ALL BOUNDS: Solve an LAP with costs

$$\begin{aligned}
&d_{ijkl} - \alpha_{ijkl} - \sigma'_{ijk} - \theta'_{ijl} \quad \forall (k,l), i < k, l \ne j\\
&d_{ijkl} + \alpha_{klij} - \sigma'_{ijk} - \theta'_{ijl} \quad \forall (k,l), i > k, l \ne j
\end{aligned}$$

and obtain optimal dual variables σ_{ijk}, θ_{ijl}, and the slacks β_{ijkl}.

Step 1c AJB': For all (k,l), $(k > i,\ l \neq j)$, move slacks to complementary elements via

$$\alpha_{ijkl} = \alpha_{ijkl} + \beta_{ijkl}, \quad \beta_{ijkl} = 0,$$

Step 2 ALL BOUNDS: Solve an LAP with costs

$$c_{ij} + \sum_{k \neq i} (\sigma_{ijk} + \sigma'_{ijk}) + \sum_{l \neq j} (\theta_{ijl} + \theta'_{ijl}) - \lambda'_i - \mu'_j$$

and obtain optimal dual variables λ_i, μ_j, and the slacks γ_{ij}.

Step 3 ALL BOUNDS: Set $\lambda'_i = \lambda'_i + \lambda_i$, $\mu'_j = \mu'_j + \mu_j$, $\lambda_i = \mu_j = 0$, and output current bound $\sum_{i=1}^{n} \lambda'_i + \sum_{j=1}^{n} \mu'_j$.

CMB & HGB: For $1 \leq i,j,k,l \leq n$, $i \neq k$, $l \neq j$ set:

$$\begin{aligned} \sigma'_{ijk} &= \sigma'_{ijk} + \sigma_{ijk} - \gamma_{ij}/(n-1) \\ \theta'_{ijl} &= \theta'_{ijl} + \theta_{ijl} \\ \beta_{ijkl} &= \beta_{ijkl} + \gamma_{ij}/(n-1) \\ \sigma_{ijk} &= \theta_{ijk} = \gamma_{ij} = 0 \end{aligned}$$

CMB: For all (i,j,k,l) $(i < k,\ l \neq j)$ interchange the following variables: $\sigma'_{ijk} \leftrightarrow \sigma'_{kli}$, $\theta'_{ijl} \leftrightarrow \theta'_{klj}$, $\beta_{ijkl} \leftrightarrow \beta_{klij}$, $d_{ijkl} \leftrightarrow d_{klij}$.

AJB & AJB': For all i,j,k,l, $i < k$, $j \neq l$ set

$$\alpha_{ijkl} = \alpha_{ijkl} + (1 - \varphi_{ijkl})(\beta_{ijkl} + \frac{\psi_{ij}}{n-1}\gamma_{ij}) - \varphi_{ijkl}(\beta_{klij} + \frac{\psi_{kl}}{n-1}\gamma_{kl}).$$

ALL BOUNDS: Set $\lambda'_i = \mu'_j = 0$ for all $1 \leq i,j \leq n$, and $s = s + 1$. If $s \leq$ max_it goto Step 1, otherwise STOP.

In the remainder of this subsection we shortly describe what the different steps of our algorithm do.

In Step 0 we set all dual variables to 0, expect for β_{ijkl} and γ_{ij} which are set to d_{ijkl} and c_{ij}, respectively. This setting clearly yields a feasible solution for DCLP. Then for all bounds but the HGB the substeps of Step 1 are performed in lexicographic order. In the case of HGB, different orders for the processing of block LAPs through the substeps of Step 1 have been tested. In the first iteration the lexicographic order is used. In each of the next iterations the first block LAPs to be processed are those whose leaders have been equal to zero at the end of the last iteration. Further, different processing orders have been proposed and tested for these block LAPs, and also for the remaining ones, i.e. those whose leaders are *different* from zero at the end of the previous iteration. In the last section of the paper we give more information on different processing orders within these two subclasses of block LAPs, and the impact they have on the quality of the produced bounds.

In Step 1a which is performed only in the case of HGB we merge the complementary elements. This means that for the current fixed pair (i,j) all costs in entry (k,l) are updated by adding to them the cost in entry (i,j) of block (k,l). We distinguish two cases $i<k$ and $i>k$ because the variables α_{ijkl} are only defined for $i<k$, $j\neq l$. These two cases, i.e. $i<k$ and $i>k$ are distinguished throughout the rest of the algorithm.

In Step 1b which is common for all bounds, a block LAP for the fixed pair (i,j) is solved. Notice that when performing this step we do not have a feasible solution for DCLP because we solve here only the block LAPs whereas the leader LAP will be solved in Step 2.

In Step 1c which is performed only in the case of AJB', we move the reduced costs β_{ijkl} (which arise after solving the LAP in block (i,j)) from the current block (i,j) to the respective complementary elements, and then set the reduced costs in block (i,j) to 0. This move is performed by means of the variables α_{ijkl} which are included in the costs of the block LAPs solved in Step 1b. Notice that after these updates, the constraints of the first and second group in DCLP remain fulfilled for all (i',j') preceding the current (i,j) according to the lexicographic order.

In Step 2 which is common to all bounds, the leader LAP is solved. Its costs are given as sum of the current reduced costs $c_{ij}-\lambda'_i-\mu'_j$ and the optimal values of the block LAPs. In this way we complete a Gilmore-Lawler like computation.

In Step 3 we output the current bound in terms of the optimal dual variables of the last LAP we solved (the variables λ', μ'). Notice that this is a valid lower bound for CLP, and hence for the QAP, because at this moment we have a feasible solution of DCLP. Indeed, for each pair (i,j) we have fixed α_{ijkl} and have computed σ_{ijk}, θ_{ijl} and β_{ijkl}, for all $k\neq i$ and $l\neq j$, so as to satisfy the two first groups of constraints in DCLP (Step 1b). Then, with these σ_{ijk} and θ_{ijl} we compute λ_i, μ_j and γ_{ij}, for all (i,j), so as to satisfy the third group of constraints in DCLP. As the two first groups of constraints do not involve the variables λ_i, μ_j and γ_{ij} computed later, those constraints remain satisfied and we have a feasible solution of DCLP.

Then we update the variables with values to be involved in the new feasible solution of DCLP.

In the case of CMB and HGB we uniformly distribute the slacks γ_{ij} to the reduced costs of the block-LAP LAP$^{(i,j)}$. This is done by means of the variables β_{ijkl} and σ'_{ijk}. These variables are involved directly (the σ-s) und indirectly (the β-s) in the costs of the block-LAPs to be solved in the next iteration (Step 1b). The fact that we subtract a portion of the slack γ_{ij} (equal to $\frac{1}{n-1}\gamma_{ij}$ form the variables σ'_{ijk} means that we are doing a row-wise distribution. One could do a column-wise distribution as well, by subtracting that portion of the slack from the variables θ'_{ijl}. After the distribution of γ_{ij} we set these variables to 0 to obtain feasibility. For the same reason we set σ_{ijk} and θ_{ijl} to 0 (consider the first and second group of constraints in DCLP). In the case of CMB we additionally do a complete exchange of the complementary elements, which is performed by a complete swap of variables involved in the two first groups of constraints of DCLP. The third group of constraints of DCLP is not necessarily fulfilled at this point. This however does not matter because we need feasibility of DCLP only at Step 3 when we output the lower bound. Feasibility is reestablished after Step 2 of the next iteration, where λ_i, μ_j and γ_{ij} will be set so as to satisfy the third group of constraints of DCLP.

In the case of AJB and AJB' we update the variables α_{ijkl} (which correspond to the Lagrange multipliers in the Lagrangian dual CLP$_\alpha$) exactly as in [1].

Finally, for all bounds we set $\lambda'_i=\mu'_j=0$ for all (i,j). Then, we increase the iteration counter by one and repeat the whole procedure for a prespecified number of iterations.

It remains only to explain why we set $\lambda'_i = \mu'_j = 0$. We do so because otherwise we would have to work with the reduced costs of the leader matrix $(c_{ij} - \lambda'_i - \mu'_j)$ in Step 2. If we did that we would have to work with an additive bound, in the sense that we would start with a bound equal to 0 and would add some nonnegative amount to it in each iteration (see e.g. how Algorithm 3.2 does). But at this point we prefer to have a non-additive bound because in the original description all bounds except for HGB are not additive. However, we also obtain the additive bound HGB in the frame of Algorithm 3.1. Indeed, assume for simplicity that he have performed two iterations of Algorithm 3.1 with specifications of HGB. If we did not set the variables λ'_i and μ'_j to 0 we would solve the leader LAP of Step 2 and add its optimal value to the current HGB as Hahn and Grant do in their original paper (see [13]). Let us denote by $\bar{c}_{ij}$ and by $\text{opt}(\bar{c})$ the costs and the optimal value of this LAP, respectively. Further, at the end of the first iteration HGB would be equal to $\sum_i \lambda'_i + \sum_j \mu'_j$. Thus, in the additive version we would have $\text{HGB} = \text{opt}(\bar{c}) + \sum_i \lambda'_i + \sum_j \mu'_j$ after the second iteration. Now what happens when we set $\lambda'_j = \mu'_j = 0$? Notice that this setting is equivalent to adding the optimal values of the dual variables λ'_i, μ'_j to the costs $\bar{c}_{ij}$. Hence, in Step 2 we actually solve an LAP with costs $c_{ij} + \lambda'_i + \mu'_j$. But an LAP with costs $\lambda'_i + \mu'_j$ is a constant problem, in the sense that all feasible assignments yield the same value of the objective function. From this fact it follows that the optimal value of the LAP with costs $c_{ij} + \lambda_i + \mu_j$ equals $opt(\bar{c}) + \sum_i \lambda'_i + \sum_j \mu'_j$, which is equal to the additive bound we would obtain in the fashion of Hahn and Grant. Thus, we obtain HGB by Algorithm 3.1.

Above we argued that in each iteration the bound output in Step 3 is the objective function value corresponding to a feasible solution of DCLP. This proves the correctness of the algorithm, in the sense that it produces a lower bound for the QAP. Additionally, it is not difficult to see that CMB, AJB, AJB', HGB are obtained by performing the corresponding steps of Algorithm 3.1. One just has to compare the bounding procedures described in Section 2 step by step to the specific steps of Algorithm 3.1. Here the only non-trivial point is that of the additive bound HGB, which was explained in detail above. An illustrative example is described in detail in the appendix.

To conclude this subsection notice that in the case of GLB, CMB and AJB the block LAPs can be processed in any arbitrary order. The values of these bounds are independent of the order. As for HGB and AJB', also in this case we may proceed the block LAPs in any order, but due to interaction between complementary elements the values of the bounds would depend on the order. Moreover, in the case of AJB' we would have to modify or Step 1.c appropriately in order to obtain a valid lower bound. This is due to the fact that after having processing a block LAP in block (i,j), we may only modify the α-s of the $\text{LAP}^{(i',j')}$ to be processed after (i,j).

3.2 A general dual framework

In this subsection we describe how different steps of Algorithm 3.1 can be extended and generalized. This extension shows where the various bounding techniques can be improved, and how HGB' can be derived. Moreover, the general dual framework can then also be viewed as homogeneous description of bounds based on linearization.

One place for generalization and extension is the interaction between complementary elements: HGB/Step 1a, AJB'/Step 1c, AJB & AJB' & CMB/Step 3 in Algorithm 3.1. In the general dual framework, which is given in Algorithm 3.2, all interactions between complementary elements are contained in Step 1a (Costs from complementary elements).

Two other extensions can be achieved in generalizing the redistribution of the leader elements: AJB & AJB' & CMB & HGB/Step 3 in Algorithm 3.1), and in a general scheme for the redistribution of parts of the bound as proposed by Hahn and Grant for HGB'. In Algorithm 3.2, the redistribution of parts of leaders is described in Step 3c (Redistribution of leaders) and the redistribution of parts of the bound is modeled in Step 3b (Redistribution of bound). The redistribution of the bound in HGB' is then a special case of Step 3b in Algorithm 3.2, as shown below.

First, we state the general dual framework, and then we describe each step of the algorithm in detail. The general algorithm contains various parameters, which allow to model the different bounds together and which also show the differences between them. We use a flag F_{ijkl}, parameters ω_{ijkl} and τ_{ijkl} to model a more general interaction between complementary costs, parameters η_i, χ_j, and ρ to model the redistribution of parts of the bound, and parameters δ_{ijk}, ε_{ijl}, and κ_{ij} to model the redistribution of parts of the leaders.

Algorithm 3.2 (General Algorithm)

Step 0 (Initialization)
Input the size of the problem n, costs D and C and maximal number of iterations max_it. Initialize $\alpha_{ijkl} = \sigma_{ijk} = \sigma'_{ijk} = \theta_{ijl} = \theta'_{ijl} = \lambda_i = \lambda'_i = \mu_j = \mu'_j = F_{ijkl} = b = 0$, $\beta_{ijkl} = d_{ijkl}$, and $\gamma_{ij} = c_{ij}$. Set iteration counter $s = 1$.

Step 1 (Block loop)
For all pairs (i,j) (in some particular order), do the following substeps of Step 1.

Step 1a (Costs from complementary elements)
For all (k,l) $(k > i,\ l \neq j)$, choose ω_{ijkl} and τ_{ijkl} such the $0 \leq \omega_{ijkl}, \tau_{ijkl} \leq 1$. Set

$$\left.\begin{aligned} \alpha_{ijkl} &= \alpha_{ijkl} + \omega_{ijkl}\beta_{ijkl} - \tau_{ijkl}\beta_{klij} \\ \beta_{ijkl} &= (1-\omega_{ijkl})\beta_{ijkl} + \tau_{ijkl}\beta_{klij} \\ \beta_{klij} &= \omega_{ijkl}\beta_{ijkl} + (1-\tau_{ijkl})\beta_{klij} \end{aligned}\right\} \quad \text{if } i < k$$

$$\left.\begin{aligned} \alpha_{klij} &= \alpha_{klij} + \omega_{klij}\beta_{klij} - \tau_{klij}\beta_{ijkl} \\ \beta_{ijkl} &= \omega_{klij}\beta_{klij} + (1-\tau_{klij})\beta_{ijkl} \\ \beta_{klij} &= (1-\omega_{klij})\beta_{klij} + \tau_{klij}\beta_{ijkl} \end{aligned}\right\} \quad \text{if } i > k.$$

Set $F_{ijkl} = 1$.

Step 1b (Block LAP)
Solve LAP$^{(i,j)}$ with costs

$$d_{ijkl} - \alpha_{ijkl} - \sigma'_{ikl} - \theta'_{ijl} \quad \forall (k,l), i < k, l \neq j$$

$$d_{ijkl} + \alpha_{klij} - \sigma'_{ikl} - \theta'_{ijl} \quad \forall (k,l), i > k, l \neq j$$

and obtain optimal dual variables σ_{ijk}, θ_{ijl}, and the slacks β_{ijkl}.

Step 2 (Leader LAP)
Solve l-LAP with costs

$$c_{ij} + \sum_{k \neq i}(\sigma_{ijk} + \sigma'_{ijk}) + \sum_{l \neq j}(\theta_{ijl} + \theta'_{ijl}) - \lambda'_i - \mu'_j$$

and obtain optimal dual variables λ_i, μ_j, and the slacks γ_{ij}.

Step 3a (Update of bound)
Set $\sigma'_{ijk} = \sigma'_{ijk} + \sigma_{ijk}$, $\theta'_{ijl} = \theta'_{ijl} + \theta_{ijl}$, $\lambda'_i = \lambda'_i + \lambda_i$, $\mu'_j = \mu'_j + \mu_j$, $\sigma_{ijk} = \theta_{ijl} = \lambda_i = \mu_j = 0$, and update bound

$$b = b + \sum_{i=1}^{n} \lambda'_i + \sum_{j=1}^{n} \mu'_j.$$

Output current bound b. Set $s = s + 1$. If $s >$max_it then STOP.

Step 3b (Redistribution of bound)
Choose η_i, χ_j, and ρ, such that

$$\sum_i \eta_i + \sum_j \chi_j + \rho = 1, \text{ and } \eta_i, \chi_j, \rho \geq 0.$$

Set

$$\lambda'_i = \lambda'_i - \eta_i b, \mu'_j = \mu'_j - \chi_j b, \gamma_{ij} = \gamma_{ij} + (\eta_i + \chi_j) b,$$

and $b = \rho b$.

Step 3c (Redistribution of leaders)
Choose δ_{ijk}, ε_{ijl}, and κ_{ij}, such that

$$\sum_{k \neq i} \delta_{ijk} + \sum_{l \neq j} \varepsilon_{ijl} + \kappa_{ij} = 1, \text{ and } \delta_{ijk}, \varepsilon_{ijl}, \kappa_{ij} \geq 0.$$

Set

$$\sigma'_{ijk} = \sigma'_{ijk} - \delta_{ijk}\gamma_{ij}, \theta'_{ijl} = \theta'_{ijl} - \varepsilon_{ijl}\gamma_{ij}, \beta_{ijkl} = \beta_{ijkl} + (\delta_{ijk} + \varepsilon_{ijl})\gamma_{ij},$$

and $\gamma_{ij} = \kappa_{ij}\gamma_{ij}$.

Next we describe each step of the Algorithm in detail. Within the description of the different steps we will also point out, how the bounds described in Section 2 can be obtained. We do this by referring to Table 1 which contains the values for the parameters ω_{ijkl}, τ_{ijkl}, η_i, χ_j, ρ, δ_{ijk}, ε_{ijl}, and κ_{ij} producing the different bounds.

Step 0 does not change, except for the initialization of flags F_{ijkl}, which are used in Step 1a of Algorithm 3.2.

In Step 1a, the exchange of costs between a pair of complementary elements is extended in the following way. We can choose any combination of parts from the current complementary costs in (i,j,k,l) and (k,l,i,j), determined by parameters ω_{ijkl} and τ_{ijkl}. The only restriction on these parameters is that they have to be nonnegative and bounded from above by 1. For $i < k$ the variable ω_{ijkl} determines the portion of β_{ijkl} which is moved to element (k,l,i,j), while τ_{ijkl} determines the portion of β_{klij} added to (i,j,k,l). The setting of a flag F_{ijkl} allows to recognize, whether an element or its complementary element have already been processed. The updates of β_{ijkl} and β_{klij} maintain feasibility.

For GLB, there is no interaction between the complementary pair, hence $\omega_{ijkl} = 0$ and $\tau_{ijkl} = 0$ as given in Table 1.

In the case of CMB, the first iteration $(s = 1)$ corresponds to GLB, i.e. there is no interaction. In the following iterations, the costs in the complementary pair are swapped before one of them is processed, thus $\omega_{ijkl} = 1$ and $\tau_{ijkl} = 1$ if $F_{klij} = 0$.

For AJB and AJB', we postpone the explanation of the choice of the parameters and their relation to the update formula (16) of Adams and Johnson to Subsection 3.3 and only describe the update within the general dual framework. In AJB and AJB', the first iteration also corresponds to computing GLB. Then for AJB, the costs within a pair of complementary elements are equally distributed before any of the two elements is processed, while for AJB', slack is moved to the complementary element which is modeled as getting the costs from (k, l, i, j) if $F_{klij} = 1$.

When computing HGB and HGB', there is always interaction between a pair of complementary elements. In the first case, the entire costs are added to the current element, while in HGB', the first update adds 50% of the complementary costs and moves 50% of the current costs to them.

Steps 1b and 2 of Algorithm 3.2, the solution of block and leader LAPs, is equivalent to Steps 1b and 2 in Algorithm 3.1. Step 3a, the update of the bound, does the same as the first part of Step 3 in Algorithm 3.1. The only difference is that the bound here is additive. This corresponds to the fact that we do not set $\lambda'_i = \mu'_j = 0$ at the end of each iteration (see the discussion concerning the additive bound in the previous section). These steps are performed by all the bounds under consideration.

In Step 3b, parts of the bound are redistributed to the costs of the problem. The parameters η_i, χ_j, and ρ determine how and which portions of the bound are redistributed to the leader elements. η_i and χ_j give the portion of the bound distributed to row i and column j, respectively. ρ determines the part which is not distributed. All these values must be nonnegative, and the sum of the distributed and the remaining part must be equal to the original bound, which is assured by

$$\sum_i \eta_i + \sum_j \chi_j + \rho = 1.$$

In all bounds except HGB', no redistribution of parts of the bound takes place, hence $\rho = 1$ for GLB, CMB, AJB, AJB', and HGB.

In HGB', parts of the bound are redistributed over rows after each iteration, and the distributed amount is decreased from iteration to iteration. This means that η_i are nonnegative numbers which also define ρ, the portion of the bound remaining. In the case of HGB' ρ increases from iteration to iteration.

Step 3c distributes positive slacks from the leaders to the elements in the block belonging to the leader. Again, this redistribution can be done via rows and columns, and parts of the slack can remain in the leader element. The amounts of γ_{ij} distributed to rows and columns are δ_{ijk} and ε_{ijl}, respectively, the portion not distributed is κ_{ij}.

In CMB, HGB and HGB', the entire slack is uniformly distributed to the rows of the submatrices, while in AJB and AJB' a small portion of the slack remains undistributed, i.e. $\kappa_{ij} = \frac{1}{n}$.

3.3 A comparison with the dual ascent procedures of Adams and Johnson

In this subsection we compare the dual procedures of [1] with the general dual framework introduced above.

First, we describe the update formula for AJB and AJB' within the dual framework. Without iteration superscripts, (16) is just

$$\alpha_{ijkl} = \alpha_{ijkl} + (1 - \varphi_{ijkl})[\beta_{ijkl} + \frac{\psi_{ij}}{n-1}\gamma_{ij}] - \varphi_{ijkl}[\beta_{klij} + \frac{\psi_{kl}}{n-1}\gamma_{kl}]$$

GLB	$\left[\begin{array}{l} \omega_{ijkl}=0, \tau_{ijkl}=0 \\ \eta_i=0, \chi_j=0, \rho=1 \\ \delta_{ijk}=0, \varepsilon_{ijl}=0, \kappa_{ij}=1 \end{array}\right.$
CMB	$\left[\begin{array}{l} \begin{cases} \omega_{ijkl}=0, \tau_{ijkl}=0, & \text{if } s=1 \\ \omega_{ijkl}=1, \tau_{ijkl}=1, & \text{if } s>1 \text{ and } F_{klij}=0 \\ \omega_{ijkl}=0, \tau_{ijkl}=0, & \text{if } s>1 \text{ and } F_{klij}=1 \end{cases} \\ \eta_i=0, \chi_j=0, \rho=1 \\ \delta_{ijk}=\frac{1}{n-1}, \varepsilon_{ijl}=0, \kappa_{ij}=0 \end{array}\right.$
AJB	$\left[\begin{array}{l} \begin{cases} \omega_{ijkl}=0, \tau_{ijkl}=0, & \text{if } s=1 \\ \omega_{ijkl}=\frac{1}{2}, \tau_{ijkl}=\frac{1}{2}, & \text{if } s>1 \text{ and } F_{klij}=0 \\ \omega_{ijkl}=0, \tau_{ijkl}=0, & \text{if } s>1 \text{ and } F_{klij}=1 \end{cases} \\ \eta_i=0, \chi_j=0, \rho=1 \\ \delta_{ijk}=\frac{1}{n}, \varepsilon_{ijl}=0, \kappa_{ij}=\frac{1}{n} \end{array}\right.$
AJB'	$\left[\begin{array}{l} \begin{cases} \omega_{ijkl}=0, \tau_{ijkl}=0, & \text{if } s=1 \text{ and } F_{klij}=0 \\ \omega_{ijkl}=\frac{1}{2}, \tau_{ijkl}=\frac{1}{2}, & \text{if } s>1 \text{ and } F_{klij}=0 \\ \omega_{ijkl}=0, \tau_{ijkl}=1, & \text{if } s\geq 1 \text{ and } F_{klij}=1 \end{cases} \\ \eta_i=0, \chi_j=0, \rho=1 \\ \delta_{ijk}=\frac{1}{n}, \varepsilon_{ijl}=0, \kappa_{ij}=\frac{1}{n} \end{array}\right.$
HGB	$\left[\begin{array}{l} \omega_{ijkl}=0, \tau_{ijkl}=1 \\ \eta_i=0, \chi_j=0, \rho=1 \\ \delta_{ijk}=\frac{1}{n-1}, \varepsilon_{ijl}=0, \kappa_{ij}=0 \end{array}\right.$
HGB'	$\left[\begin{array}{l} \begin{cases} \omega_{ijkl}=\frac{1}{2}, \tau_{ijkl}=\frac{1}{2}, & \text{if } F_{klij}=0 \\ \omega_{ijkl}=0, \tau_{ijkl}=1, & \text{if } F_{klij}=1 \end{cases} \\ \eta_i\geq 0, \chi_j=0, \rho=1-\sum_i \eta_i \\ \delta_{ijk}=\frac{1}{n-1}, \varepsilon_{ijl}=0, \kappa_{ij}=0 \end{array}\right.$

Table 1: Parameters for different bounds in the dual framework

for any given φ_{ijkl} and ψ_{ij}. In the above description of AJB', we use the same values as proposed in [1], namely $\varphi_{ijkl} = 0.5$ and $\psi_{ij} = \frac{n-1}{n}$. Using these values, we obtain as update

$$\alpha_{ijkl} = \alpha_{ijkl} + \frac{1}{2}[\beta_{ijkl} + \frac{1}{n}\gamma_{ij}] - \frac{1}{2}[\beta_{klij} + \frac{1}{n}\gamma_{kl}].$$

But now it is easy to see, that the update corresponds to a uniform distribution of a portion of $\frac{n-1}{n}$ of the leaders to the entries of the cost matrices of the corresponding block LAPs, with an ensuing equal distribution of costs in complementary pairs. This is also obtained using the setting given in Table 1.

From this viewpoint, the similarities and differences between AJB and AJB' and the bounds CMB, HGB, and HGB' can be readily seen. For each pair of complementary elements, costs are transferred at different times and in different rates. With respect to the distribution of positive slack from the reduced leader matrix to the cost matrices of the corresponding block LAPs, all bounds use a uniform distribution, where AJB and AJB' perform an incomplete distribution and GLB a "zero"-distribution.

In general, one can choose any φ_{ijkl} and ψ_{ij} within the update formula, see [1]. When restricted to

$$0 \leq \varphi_{ijkl} \leq 1 \text{ and } 0 \leq \psi_{ij} \leq 1,$$

Adams and Johnson show, that their dual ascent procedures produce a nondecreasing sequence of lower bounds.

In our framework, this corresponds to

$$\omega_{ijkl} = (1 - \varphi_{ijkl}) \text{ and } \tau_{ijkl} = \varphi_{ijkl},$$

or equivalently

$$\omega_{ijkl} = 1 - \tau_{ijkl},$$

which turns out to be a restriction in the choice of the ω-s and the τ-s. The only bounds other than AJB and AJB', which satisfies this restricted update are HGB and HGB', implying that HGB can be obtained via DAP (EDAP). HGB' can not be obtained by DAP (EDAP) because in the case of HGB' we have a redistribution of the bound and this is not the case in the procedures of Adams and Johnson.

Note also that the redistribution of the positive slacks in the leader matrix is more restrictive in [1], where only a uniform distribution is possible. We summarize Table 1 and the above considerations in the following observation.

Observation 3.3 *The dual ascent procedures of Adams and Johnson are a special case of the general dual framework given by Algorithm 3.2. GLB, AJB, AJB' and HGB can be obtained using these dual ascent procedures, while CMB and HGB' can not.*

Finally we give a sufficient condition on the parameters of the general dual framework, under which we obtain a nondecreasing sequence of lower bounds. This generalizes the monotonicity result of Adams and Johnson.

Theorem 3.4 *The general dual framework given by Algorithm 3.2 produces a sequence of lower bounds for the quadratic assignment problem which is nondecreasing if $\eta_i = 0$, $\chi_j = 0$, and $\rho = 1$, i.e. no portion of the bound is redistributed.*

Proof. The proof reduces to observe that both in Step 1b and in Step 2 of Algorithm 3.2 only LAPs with nonnegative costs are solved. This follows from the considerations in Subsections 3.1 and 3.2. Hence, in every update of bound b we add a nonnegative amount to it, which means that the sequence of bounds is nondecreasing. □

Note, that in general a redistribution of the bound can lead to non-monotonicity.

4 Computational experiments

In this section we propose a new parameter setting and compare the resulting dual bound with the bounds proposed in the literature. The section is organized as follows. First, we introduce the new bound and discuss its characteristics compared to the other linearization bounds described in this paper. The next subsection outlines implementational aspects and gives a short description of QAPpack, a software package based on the dual framework, see [10]. Then we report on extensive experiments which compare KCCEB with CMB, AJB, AJB', HGB, and HGB'. We also compute the bound for all instances of QAPLIB [4] with $n \leq 72$. Finally, we point out various properties of our new bound related to the application in Branch-and-Bound.

4.1 A new dual bound for QAP

Our goal is to find a setting for the parameters ω, τ, η, χ, ρ, δ, ε, and κ in Algorithm 3.2 and an order to process the block LAPs (see Step 1 of Algorithm 3.2), which a) produces a bound which is competitive with the existing bounds based on linearization in the long run, and b) shows a moderately steep increase as the number of iterations increases. Property b) means that we can obtain a relatively good value of the bound after a small number of iterations. Because of the trade off between quality of the bound and computation time such a behavior would be relevant for bounds used within Branch-and-Bound schemes (see also Section 4.4). Since the best bound of Hahn and Grant [13] is competitive with CMB, AJB, and AJB' both in terms of computation time and quality, we try to find a setting and ordering which compare well to HGB and HGB'.

We performed extensive experiments combining different settings for the parameters with different orders to process the LAPs. As for the values of the parameters we have focussed on tuning the parameters ω and τ (which describe the interaction between complementary elements), and ρ (which describes the amount of the current bound to be redistributed in the current iteration).

Settings for ω, τ: We have experimented with different fixed values kept constant in the course of the algorithm ranging between 0.2 and 0.8. Other tests have been made with values which decrease linearly (deterministically) with the number of iterations and we have experimented with different decreasing linear functions. Another setting we tested selects the values of the parameters at random from a prespecified range interval in each iteration. This interval may be kept unchanged in the course of the algorithm or may be changed in each iteration. The change consists in increasing the upper limit of the interval linearly and we have tested different linear functions for this purpose.

Settings for ρ: We have experimented with different values of ρ that were kept constant or decreased linearly or exponentially with the number of iterations. The decreasing function may be linear or exponential, random or deterministic, the latter meaning that the coefficients which define the function are chosen as random variables or constants, respectively. Moreover a number of experiments have been made to determine the "best" decreasing function both in the random and in the deterministic setting.

As for the order of solution for the block LAPs we have compared the order proposed by Hahn and Grant [13] which we call *prioritizing order*, with other orders introduced by us like the *sorting* order, several *greedy* orders and *reversed* prioritizing or sorting orders. The sorting order is a refinement of the prioritizing order. According to the sorting order, the block LAPs whose leader elements have been zero at the end of the last iteration (0-blocks), are solved at first. Then the other block LAPs are solved in non-decreasing order of the values of their leader elements (resulting at the end of the last iteration). The greedy orders are a further refinement of the sorting order. A greedy order renders more precise the order in which the 0-blocks are processed in the first phase of a sorting order. The greedy order which resulted to be the best one and which was then implemented in our new bound is the following. Process first the 0-blocks and then the other blocks like in the sorting order. Among the 0-blocks, process at first those blocks with most zeroes in their own row/column of the reduced leader matrix.

The best combination of parameter setting and processing order we obtained by our experiments is the following. We have $\omega_{ijkl} = 0.5 + z$, $\tau_{ijkl} = 0.5 + z$ if $F_{klij} = 0$, and $\omega_{ijkl} = 1$, $\tau_{ijkl} = 0$ if $F_{klij} = 1$, where z is drawn uniformly from $[0, (0.3 - x)]$ with $x = 0.3$ at iteration 0 and is multiplied by y, $0 < y < 1$, at each iteration. For the redistribution of parts of the bound, ρ is initially set to 14/15 and is then decreased by a factor of 2/3 in each 100-th iteration. This resembles the redistribution scheme proposed by Hahn and Grant [13]. Let m_i^r (m_j^c) be the number of zeroes in the i-th row (j-th column) of the reduced leader matrix, and let m be the number of zeroes of the reduced leader matrix. We set $\eta_i = m_i^r \frac{1-\rho}{2m}$ and $\chi_j = m_j^c \frac{1-\rho}{2m}$. For the redistribution of the leaders we set $\kappa = 0$, i.e. we redistribute the entire leader element. Let m_{ijk}^r (m_{ijl}^c) be the number of zeroes in row k (column l) of the reduced block matrix, and let m_{ij} be the number of all zeroes of the reduced block matrix in block (i, j). We set $\delta_{ijk} = m_{ijk}^r \frac{1}{2m_{ij}}$, $\varepsilon_{ijl} = m_{ijl}^c \frac{1}{2m_{ij}}$. These choices for η, χ, δ and ε put larger portions of the bound/leaders in rows/columns having more zeroes. This parameter setting is shown in Table 2. The new bound obtained by Algorithm 3.2 with the above setting and the greedy ordering described above is denoted by KCCEB.

KCCEB	$\left[\begin{array}{l} \left\{\begin{array}{ll} \omega_{ijkl} = \frac{1}{2} + z, \tau_{ijkl} = \frac{1}{2} + z, & \text{if } F_{klij} = 0 \\ \omega_{ijkl} = 0, \tau_{ijkl} = 1, & \text{if } F_{klij} = 1 \end{array}\right. \\ \eta_i = m_i^r \frac{1-\rho}{2m}, \chi_j = m_j^c \frac{1-\rho}{2m}, \rho \geq 0 \\ \delta_{ijk} = m_{ijk}^r \frac{1}{2m_{ij}}, \varepsilon_{ijl} = m_{ijl}^c \frac{1}{2m_{ij}}, \kappa_{ij} = 0 \end{array}\right.$

Table 2: Parameters for new bound KCCEB

4.2 Implementational Aspects

The numerical results were obtained using a Java-based implementation of KCCEB. The implementations of KCCEB and the other linearization bounds led to the software package QAPpack [10]. The package is mainly based on the dual framework described in this paper and is designed to provide a flexible framework for experimenting with different bounding methods for QAP. In QAPpack, the implementation of the dual framework is embedded in a Branch-and-Bound scheme, where the user can experiment with different settings as well.

The instances of size less than or equal to 40, were completed on a 200MHz AMD K6 with 64MB RAM, running Windows 95, which allowed these instances to fit in main memory. The larger instances were executed on a HP 9000/700 running Unix. On both architectures (AMD K6 as well as HP 9000/700) the Java code was compiled to native code to improve performance. As shown in performance comparisons of QAPpack [10] this approach produces performance similar to that of compiled C-code.

Since KCCEB (as well as AJB, AJB', HGB and HGB') operates on fractions of costs, and reuses these fractions in several iterations, it was necessary to use fixed point arithmetic in the implementation, to avoid accumulation of floating point roundoff errors. The fixed point approach gives on one hand absolute accuracy, but on the other hand can not guarantee exact equal redistribution of fractions of leader elements. This however does not seem to have any significant effect.

We were not able to reproduce the best values for the bound obtained by Hahn and Grant [13] although we tried all settings described in [13]. In the case of randomly selected values for the parameters, we performed a large number of tests in order to obtain representative results. The best results we obtained with a parameter setting in the fashion of Hahn and Grant are slightly poorer than those reported in [13]. To obtain these results ω and τ were kept constant, $\omega = 0.2$, $\tau = 0.8$, ρ was initially set to 14/15 and was then decreased by a factor of 2/3 in each 100-th iteration, $\eta = \frac{1-\rho}{n}$, $\chi = 0$, and $\kappa = 1$, $\delta = 0$, $\varepsilon = 0$. The bounds obtained by this parameter setting are denoted by HGB**, while the bounds given in Table III in [13] are referred to as HGB*.

4.3 Numerical results

In this section we report on our numerical results concerning the comparison of our new bound KCCEB with other linearization bounds, especially with the bounds CMB, AJB, AJB', HGB and HGB' obtained in our dual framework. More precisely we address the following issues: the number of iterations needed to make a fair comparison of linearization bounds, the comparison of the existing linearization bounds, the comparison of the new bound KCCEB with the best among the existing linearization bounds, the comparison of KCCEB with the bound RRDB of Resende, Ramakrishnan and Drezner [20] obtained by solving CLP – the continuous linear relaxation of QAP to optimality – and finally, the performance of the new bound in terms of computation time.

Number of iterations and comparison of existing linearization bounds. 2000 iterations of CMB, AJB, AJB', HGB and HGB' are run on three instances Had20, Nug24 and Sko42 from QAPLIB in order to evaluate the evolution of each of these bounds.

Figure 1 shows the evolution of the bounds CMB, AJB, AJB', HGB, and HGB' with the number of iterations. The values correspond to the bounds obtained after 2^k ($k \in \{0, \ldots, 10\}$) and 2000 iterations. CMB, AJB, and AJB' do not improve significantly when the number of iterations exceeds 256. HGB improves a little but is poorer than CMB, AJB, and AJB'. HGB' improves slightly when the number of iterations exceeds 256 but is already superior to the other bounds by 256 iterations. Based on these results we conclude that HGB' is superior to the other existing linearization bounds already by 256 iterations, confirming in this way results existing in the literature (see [13]). Moreover according to Figure 1 we can conclude that 256

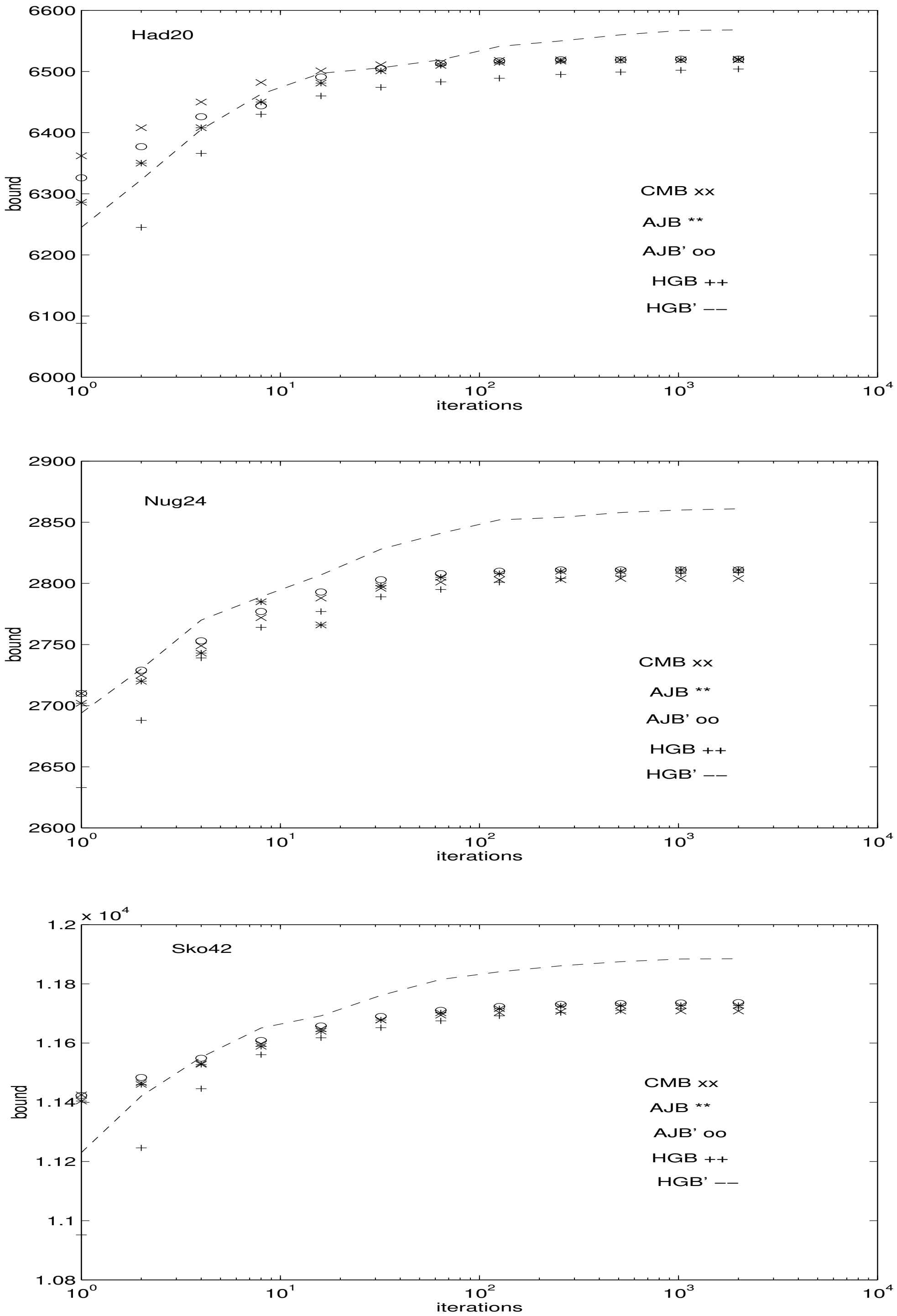

Figure 1: Evolution of existing linearization bounds

iterations are enough to evaluate and/or compare the behavior of the existing bounds. In the following we show that KCCEB competes well with HGB', thus showing that KCCEB is also superior to the other linearization bounds.

Comparison of KCCEB with HGB and RRDB.** As mentioned in the previous section we could not reproduce the values HGB* reported in Table III in [13]. However, as the following results show, our implementation HGB** of the bounding scheme of Hahn and Grant is competitive with or only slightly poorer than HGB* for any number of iterations until 2000.

In Table 3 HGB*, HGB** and KCCEB are compared on several test instances from QAPLIB. In the case that KCCEB is better than HGB* (HGB**) the corresponding entries are printed in bold face.

Bound\Inst	Nug12	Nug15	Nug20	Rou20	Kra30a	Kra30b	Nug30
HGB*(100)	517	1029	2159	636036	74934	75635	4745
HGB**(128)	518	1031	2165	636207	75252	76076	4765
KCCEB(128)	**520**	1026	**2167**	**640487**	**75217**	76064	**4723**
HGB**(256)	518	1034	2168	638185	75529	76261	4775
KCCEB(256)	**521**	1032	**2170**	**641224**	**75546**	**76294**	**4783**
HGB*(500)	522	1038	2175	640459	75679	76375	4780
HGB**(512)	521	1034	2170	640322	75580	76221	4776
KCCEB(512)	**522**	1036	2174	**641933**	**75699**	**76430**	**4791**
HGB*(2000)	523	1040	2177	641663	75854	76563	4789
HGB**(2000)	522	1036	2173	641192	75714	76373	4783
KCCEB(2000)	522	1037	2176	**642479**	75792	76517	**4795**

Table 3: A comparison of HGB*, HGB** and KCCEB

Table 3 shows that KCCEB competes well with HGB* (HGB**). In particular, in the short run KCEEB is generally better than HGB* (HGB**), while it achieves a value which is close to that of HGB* (HGB**) in the long run.

Further we compare the new bound KCCEB with the bound RRDB obtained by solving the continuous relaxation CLP of LP by a dual interior point method, see Tables 4 and 5. Clearly RRDB is better than KCCEB, because any bound obtained by Algorithm 3.2 is a lower bound to the optimal solution of the continuous relaxation of LP. However RRDB is well known to be a very expensive bound in terms of computation time, and hence practically infeasible for use within Branch-and-Bound algorithms (see [20] and also the discussion of computation time issues below).

Tables 4 and 5 compare the values of KCCEB obtained after 1 and after 256 iterations with RRDB, the optimal solution of CLP. In the third column, we give the best known solutions to the instances. The values marked by an "$*$" are not known to be optimal. Entries "n.a." mean that the results are not available, and "n.v." that we could not verify the value given in the literature. Finally, the last two columns show the relative gap between RRDB and KCCEB and the relative improvement in the value of KCEEB when the number of iterations increases from 1 to 256, i.e.

$$\text{gap} = \frac{\text{RRDB} - \text{KCCEB}(256)}{\text{RRDB}} * 100, \quad \text{red} = \frac{\text{KCCEB}(256) - \text{KCCEB}(1)}{\text{KCCEB}(1)} * 100.$$

Inst	size	Sol	GLB	RRDB	KCCEB(1)	KCCEB(256)	gap %	red %
Bur26a	26	5426670 *	5315200	n.v.	5317729	5356826	–	0.7
Bur26b	26	3817852 *	3714750	n.v.	3716785	3753198	–	1.0
Bur26c	26	5426795 *	5312038	n.v.	5317292	5361204	–	0.8
Bur26d	26	3821225 *	3711739	n.v.	3716679	3758687	–	1.1
Bur26e	26	5386879 *	5307079	n.v.	5310637	5334780	–	0.5
Bur26f	26	3782044 *	3706888	n.v.	3710034	3733941	–	0.6
Bur26g	26	10117172 *	9978273	n.v.	9984869	10055637	–	0.7
Bur26h	26	7098658 *	6973253	n.v.	6978837	7045690	–	1.0
Chr12a	12	9552	7245	9552	7368	9448	1.1	28.2
Chr12b	12	9742	7146	9742	7318	9742	0.0	33.1
Chr12c	12	11156	7976	11156	8460	10978	1.6	29.8
Chr15a	15	9896	5625	9511	6186	8743	8.1	41.3
Chr15b	15	7990	4653	7990	4827	7036	12.0	45.8
Chr15c	15	9504	6165	9504	7041	9143	3.8	29.9
Chr18a	18	11098	6779	10752	7280	10037	6.6	37.9
Chr18b	18	1534	1534	1534	1534	1534	0.0	0.0
Chr20a	20	2192	2150	2174	2156	2166	0.4	0.5
Chr20b	20	2298	2196	2287	2236	2259	1.2	1.0
Chr20c	20	14142	8601	14142	9131	13056	7.7	43.0
Chr22a	22	6156	5924	6143	5946	6120	0.4	2.9
Chr22b	22	6194	5936	6181	5975	6154	0.4	3.0
Chr25a	25	3796	2765	3785	2846	3607	4.7	26.7
Els19	19	17212548	11971949	16874205	12701141	16274994	3.6	28.1
Esc16a	16	68	38	48	38	41	14.6	7.9
Esc16b	16	292	220	278	220	274	1.4	24.5
Esc16c	16	160	83	118	83	91	22.9	9.6
Esc16d	16	16	3	4	3	4	0.0	30.4
Esc16e	16	28	12	14	12	12	14.3	0.0
Esc16g	16	26	12	14	12	12	14.3	0.0
Esc16h	16	996	625	704	629	704	0.0	11.9
Esc16i	16	14	0	0	0	0	–	–
Esc16j	16	8	1	2	1	2	0.0	100.0
Had12	12	1652	1536	n.a.	1544	1619	–	4.9
Had14	14	2724	2492	n.a.	2511	2661	–	6.0
Had16	16	3720	3358	n.a.	3383	3553	–	5.0
Had18	18	5358	4776	n.a.	4819	5078	–	5.4
Had20	20	6922	6166	n.a.	6221	6567	–	5.6
Kra30a	30	88900 *	68360	76003	69410	75566	0.6	8.9
Kra30b	30	91420 *	69065	76752	70400	76235	0.7	8.3
Lipa20a	20	3683	3667	3683	3681	3683	0.0	0.1
Lipa30a	30	13178	13147	13178	13174	13178	0.0	0.0
Lipa40a	40	31538	31427	n.a.	31528	31538	–	0.0
Lipa50a	50	62093	61940	n.a.	62079	62093	–	0.0
Lipa60a	60	107218	107048	n.a.	107207	107207	–	0.0
Lipa70a	70	169755	169506	n.a.	169735	169735	–	0.0

Table 4: KCCEB for QAPLIB instances (Part 1)

Inst	size	Sol	GLB	RRDB	KCCEB(1)	KCCEB(256)	gap %	red %
Nug12	12	578	493	523	495	521	0.4	5.2
Nug15	15	1150	963	1041	965	1033	0.8	7.0
Nug16a	16	1610	1314	n.a.	1319	1419	–	7.6
Nug16a	16	1240	1022	n.a.	1024	1082	–	5.7
Nug17	17	1732	1388	n.a.	1390	1498	–	7.8
Nug18	18	1930	1554	n.a.	1556	1656	–	6.4
Nug20	20	2570	2057	2182	2061	2173	0.5	5.4
Nug21	21	2438	1833	n.a.	1843	2008	–	9.0
Nug22	22	3596	2483	n.a.	2515	2834	–	12.7
Nug24	24	3488	2676	n.a.	2680	2857	–	6.6
Nug25	25	3744	2869	n.a.	2877	3064	–	6.5
Nug30	30	6124 *	4539	4805	4550	4785	0.5	5.1
Rou12	12	235528	202272	224278	203994	223543	0.3	9.6
Rou15	15	354210	298548	324869	301842	323589	0.4	7.2
Rou20	20	725522	599948	643346	605335	641425	0.3	6.0
Scr12	12	31410	27858	29872	27858	29538	1.1	6.0
Scr15	15	51140	44737	49264	44758	48547	1.5	8.5
Scr20	20	110030	86766	95113	86766	94489	0.7	8.9
Sko42	42	15812 *	11311	n.a.	11328	11879	–	4.9
Sko49	49	23386 *	16161	n.a.	16205	17054	–	5.2
Sko56	56	34458 *	23321	n.a.	23358	24570	–	5.2
Sko64	64	48498 *	32522	n.a.	32572	32572	–	0.0
Sko72	72	66256 *	44280	n.a.	44331	46411	–	4.7
Ste36a	36	9526 *	7124	n.a.	7394	7860	–	6.3
Ste36b	36	15852 *	8653	n.a.	27809	32718	–	17.7
Ste36c	36	8239110 *	6393629	n.a.	32231	32751	–	1.6
Tai12a	12	224416	195918	n.a.	197891	220804	–	11.5
Tai12b	12	39464925	9788461	n.a.	20588238	30523552	–	48.3
Tai15a	15	388214	327501	n.a.	330912	351938	–	6.3
Tai15b	15	51765268	11242074	n.a.	49769648	51494600	–	3.5
Tai17a	17	491812	412722	n.a.	416368	441501	–	6.0
Tai20a	20	703482	580674	n.a.	584667	616644	–	5.4
Tai20b	20	122455319	14205869	n.a.	34654520	92528264	–	167.0
Tai25a	25	1167256 *	962417	n.a.	967236	1005978	–	4.0
Tai25b	25	344355646	47692620	n.a.	82068728	148325408	–	80.7
Tai30a	30	1818146 *	1504688	n.a.	1510823	1565313	–	3.6
Tai30b	30	637117113 *	41183334	n.a.	83287792	137224736	–	64.8
Tai35a	35	2422002 *	1951207	n.a.	1958571	2018380	–	3.0
Tai35b	35	283315445 *	30866283	n.a.	62917308	101860224	–	61.9
Tai40a	40	3139370 *	2492850	n.a.	2501800	2574810	–	2.9
Tai40b	40	637250948 *	47124379	n.a.	107433928	171497504	–	59.6
Tai50a	50	4941410 *	3854359	n.a.	3864461	3956707	–	2.3
Tai50b	50	458821517 *	40543466	n.a.	79700440	117283280	–	47.2
Tai60a	60	7208572 *	555509	n.a.	5567731	5691710	–	2.2
Tai60b	60	608215054 *	51376763	n.a.	125852736	215230576	–	71.1
Tai64c	64	1855928 *	725309	n.a.	725309	725309	–	0.0
Tho30	30	149936 *	90578	100784	91098	99855	0.9	9.6
Tho40	40	240516 *	143804	n.a.	144334	154642	–	7.1
Wil50	50	48816 *	38069	n.a.	38301	40392	–	5.4

Table 5: KCCEB for QAPLIB instances (Part 2)

For more information about the instances and for related references we refer to QAPLIB [4]. Note that the instances Lipaxxb are not included since GLB is tight for this class of problems.

We could not verify the values of RRDB for the asymmetric instances Bur26x as given by Resende et al. In [20], the values reported for GLB are obtained for costs $d_{ijkl} = a_{ik}b_{lj}$, i.e. the second matrix is transposed and this does not yield a feasible bound for the instance. We tried to approximate the value of RRDB as reported in [20] by our scheme, but KCCEB produces for all possible combinations for obtaining d_{ijkl} (including illegal ones like above) a value larger than RRDB. Therefore we refer to the corresponding entries in Table 4 as not verifiable.

Computation time. The execution times for KCCEB, as well as for the other iterative bounds computed by Algorithm 3.2, are unaffected by the instance type, and only dependent on the instance size.

The results for RRDB [20] were obtained on a Silicon Graphics Challenge with 150 MHz and 1.5 Gbytes of RAM. Disregarding the relative performance of the processors (200 MHz AMD K6 against a Silicon Graphics Challenge), and the pure integer implementation of the KCCEB, we try to compare the the running times of the RRDB with KCCEB. The performance of RRDB varies both with different instance types as well as sizes, whereas as stated above KCCEB is insensitive to instance types.

On a few special instances the performance of 256 iterations of KCCEB is actually slower than computing RRDB (i.e. Esc16d about 5.5 times slower), and on some instances KCCEB is several orders of magnitude faster (Chr25a about 148 times faster), but in general KCCEB is at least one order of magnitude faster. Table 6 gives the average running times of 256 iterations of KCCEB on the AMD K6 and the average computation time of RRDB on the Silicon Graphics Challenge for specific instance sizes.

Bound\Size	12	15	16	18	19	20	25	26	30	35	40
KCCEB	13	33	44	73	91	112	299	360	929	2094	3079
RRDB	163	956	128	2154	5228	3822	44227	6863	18153	n.a.	n.a.

Table 6: Running times (in seconds) for 256 iterations of KCCEB and for RRDB

4.4 Application in Branch-and-Bound

Traditionally, a bound function is considered to be a black box, which takes a subproblem of the search space as input and delivers a lower bound for the optimal value of the objective function as output. In this framework it makes sense to discuss the relative strength of bounding functions for the same problem.

In case a bounding function is iterative, i.e. the bound is the result of a number of iterations of some procedure, one has a whole family of bounding functions rather than one single function. The strength of one of these is clearly defined only if the stopping criterion, i.e. the condition for terminating the iterative procedure, is clearly defined, and this is often not the case.

Furthermore, it may be the case that the best bound achievable with an iterative bound using a fixed number of iterations is not obtained by iterating on the given problem, but rather by branching and computing the iterative bound in each subproblem using a substantially smaller number of iterations, and then taking the minimum of the bounds obtained. From the black box point of view, such a bound is as sound as any non-iterative bound, but when one wants to compare strength of bounds, it is not clear how to make a fair comparison.

The situation much resembles that occurring when evaluating iterative heuristics like e.g. simulated annealing. One has (at least) three parameters: quality, time, and space. Given the time available measured as the number of iterations, one has to decide on the optimal way to spend the time: One long annealing or a number of shorter ones. It is generally accepted that one experimentally finds the tuning of the algorithm with respect to length of the individual annealing. Likewise, in the following, any bound for a subproblem resulting from using a prespecified number of iterations of an iterative bounding procedure on the subproblem and some of its descendants, and then deriving a bound from these results, is regarded as a bound in the traditional sense.

For dual bounds for QAP such as the one presented in the current paper, two examples of such bounds are immediate. Suppose a subproblem is given, in which there are n free facilities to be located on n free locations, and suppose the chosen number of iterations is N. We may branch the current subproblem by either choosing one location and create a subproblem for each facility, or choosing one facility to be located on each of the free locations, in both cases creating n new subproblems. Then we bound each of these using N/n iterations. The bound is now the minimum of those of the subproblems. We may also create any combination fixing one free facility to one free location, resulting in n^2 subproblems, and use N/n^2 iterations to bound each of these. For each facility (location), the minimum bound obtained by assigning this to all free locations (facilities) is a valid bound, and since each facility (location) has to be assigned, the maximum of these minima is a valid bound. In each case the number of iterations spent in calculating the bound is N. The results may then be compared to the bound resulting from using N iterations to bound the given subproblem without branching. We refer to [7] for a more detailed discussion on these issues.

We illustrate the second idea of creating n^2 subproblems and computing KCCEB for each of those by means of the Nug20 instance where we perform 8 iterations of KCCEB per subproblem. Interpreting the bounds obtained for fixing facility i as the entries of the ith row of a matrix, and the bounds from fixing locations as the column entries, the n^2 bounds correspond to the entries of an $n \times n$ matrix. The minima over rows and columns are valid lower bounds, and for Nug20, these bounds lie in the interval $[2161, 2200]$. The maximum value in this interval is the best bound we can find with this approach using 3200 iterations overall.

However, if we just pick any row or column, i.e. choosing randomly a facility or location to branch on, and use the minimum of this as bound, we will obtain a bound of 2161 in the worst case. Moreover, it is very likely that the bound obtained by applying 8 iterations on each subproblem, i.e. 160 iterations overall, will be tighter than HGB* after 2000 iterations, which produced a bound of 2177.

For larger instances, instead of branching to generate all possible assignments, we use the branching rule utilizing information from the reduced cost obtained by computing GLB in the root node (for a description of the branching rule we refer to [9, 17]) and branch on only one facility. We also make use of symmetries in the coefficient matrices to reduce the number of subproblems considered. Taking the minimum bound of the subproblems created, we obtain a lower bound as argued above. We have computed this for several instances from QAPLIB. In order to get a fair comparison with HGB* (HGB**) we set 2000 as an upper limit for the total number of iterations of Algorithm 3.2 to be applied.

As shown by Table 7 the new bound, which we denote by KCCEB', is superior to HGB* (or HGB** in the case that HGB* is not available – entries marked by a $*$) in most of the instances we have experimented with. The only exception is Kra30b. Notice however, that we are still far below 2000 iterations here. For two instances KCCEB' gives even an improvement of the best known bound from QAPLIB (entries printed in bold face).

Inst	n	KCCEB'	Tot. no. iterations	HGB* (2000) (HGB**(2000))	
Nug12	12	530	256	523	
Nug15	15	1055	288	1040	
Had16	16	3548	384	3554	*
Had18	18	5058	448	5080	*
Nug20	20	2202	640	2177	
Rou20	20	649028	608	641663	
Had20	20	6573	1700	6568	*
Tai20a	20	627763	640	616733	*
Nug24	24	2905	736	2861	*
Tai25a	25	**1016213**	768	1006749	
Nug30	30	4816	960	4789	
Kra30a	30	**76471**	896	75854	
Kra30b	30	75772	960	76563	

Table 7: A comparison of KCCEB' and HGB*

Concluding, KCCEB' seems to be very promising. This is also supported by the findings of [8], where CMB – which is weaker than the bounds introduced here – was identified as a possible candidate for solving larger instances of QAP to optimality using Branch-and-Bound. However, the results with respect to KCCEB' are preliminary and we refer to [7] for more details.

5 Conclusions

In this paper we have proposed a dual framework to compute lower bounds for the quadratic assignment problem. The dual framework is based on an approach to compute approximate solutions to the dual of the continuous relaxation CLP of a well known mixed integer programming formulation for the QAP (the formulation LP in Section 2).

The interest in the dual framework is two-fold. From the theoretical point of view, the dual framework provides a unified representation of all existing lower bounds for the QAP based on linearization. We have shown that the bounds of Gilmore-Lawler (GLB), Carraresi and Malucelli (CMB), Adams and Johnson (AJB, AJB'), and Hahn and Grant (HGB, HGB') can be obtained by the dual framework with specific settings for the parameters. Moreover, we have shown that the differences between these bounding procedures are technical rather than principal. They concern the interaction between complementary elements, namely the rates and the time of interactions, and the rule of the re-distribution of slacks in the leader and block LAPs. As shown in Section 3.3, our approach is more general than that of Adams and Johnson [1], since the bounds CMB and HGB' can not be obtained by applying the latter, but can be obtained by applying our approach.

The dual framework is also of practical interest because it allows us to test different settings for the parameters and not only settings which are suggested by combinatorial arguments as for example in GLB, CMB or HGB. After extensive experiments we identified a parameter setting which yields a new competitive bound (KCCEB) for the QAP. The new bound has been tested and compared with the existing ones in all QAPLIB instances of size up to 72, and provides a good trade off between computation time and bound quality. The new bound is superior to most of the existing linearization bounds and it is close to the bound of Resende et al. (RRDB). Recall that RRDB is obtained by solving the continuous relaxation CLP to optimality and hence, RRDB is an upper bound for all bounds obtained by linearizations. In terms of computation time – which is a crucial issue as far as the applicability of bounds in Branch-and-Bound algorithms is concerned – KCCEB is in general at least one order of magnitude faster than RRDB.

Finally, we have discussed the efficient application of KCCEB in Branch-and-Bound algorithms. The idea is to combine bound computations with branching and exploit thereby the symmetries of the instance at hand and the information provided by the dual variables. Although the testing of such ideas is still at a preliminary stage, the bounds we obtained are very promising. We were able to improve the best existing bounds (among all bounds, not only linearization bounds) for two instances from QAPLIB, namely Tai25a and Kra30a. Furthermore, in a parallel processing environment the idea of branching early rather than computing the best possible bound in a node with a given iterative method adds substantial freedom in designing the most effective Branch-and-Bound algorithm for the QAP. Our preliminary results show that not only do we get a better bound, but as opposed to RRDB also one, for which the calculations can be easily parallelized.

Our dual framework gives a more comprehensive view of bounds based on linearization and provides a good basis for further research in two directions: computation of new bounds for the QAP and efficient application of lower bounds within Branch-and-Bound algorithms, both sequentially and in parallel. Whereas the computation of lower bounds is a well studied topic, a lot remains to be done concerning an efficient incorporation of bound computations in Branch-and-Bound algorithms for the QAP.

Acknowledgments

The first author acknowledges financial support through SNF-grant 9510057 of the Danish National Science Research Council. The second author acknowledges support by the research project Spezialforschungsbereich F003 "Optimierung und Kontrolle", Projektbereich "Diskrete Optimierung" by the Austrian Science Foundation.

We thank P. Hahn and F. Malucelli for fruitful discussions which helped us to better understand their bounding methods.

References

[1] W. P. Adams and T. Johnson, Improved linear programming based lower bounds for the quadratic assignment problem, in *Quadratic Assignment and Related Problems*, P. Pardalos and H. Wolkowicz, eds., *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **16**, 1994, pp. 43–77, AMS, Providence, Rhode Island.

[2] W. P. Adams and H. D. Sherali, A tight linearization and an algorithm for zero-one quadratic programming problems, *Management Science* **32**, 1986, 1274–1290.

[3] A. A. Assad and W. Xu, On lower bounds for a class of quadratic 0-1 programs, *Operations Research Letters* **4**, 1985, 175–180.

[4] R. E. Burkard, S. E. Karisch, and F. Rendl, QAPLIB — a quadratic assignment problem library, *Journal of Global Optimization* **10**, 1997, 391–403. The on-line version is available via: `http://opt.math.tu-graz.ac.at/~karisch/qaplib/`

[5] P. Carraresi and F. Malucelli, A reformulation scheme and new lower bounds for the QAP, in *Quadratic Assignment and Related Problems*, P. Pardalos and H. Wolkowicz, eds., *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **16**, 1994, pp. 147–160, AMS, Providence, Rhode Island.

[6] E. Çela, *The Quadratic Assignment Problem: Theory and Algorithms*, Kluwer Academic Publishers, Dordrecht, 1998.

[7] J. Clausen, S.E. Karisch, and T. Espersen, Iterative bounds in Branch-and-Bound - quality vs. time, Working Paper, Department of Mathematical Modelling, Technical University of Denmark, 1998.

[8] J. Clausen, S.E. Karisch, M. Perregaard, and F. Rendl, On the applicability of lower bounds for solving rectilinear quadratic assignment problems in parallel, *Computational Optimization and Applications*, to appear.

[9] J. Clausen and M. Perregaard, Solving large quadratic assignment problems in parallel, *Computational Optimization and Applications* **8**, 1997, 111–128.

[10] T. Espersen, S. E. Karisch, E. Çela, and J. Clausen, QAPpack – A Java package for solving quadratic assignment problems, Working Paper, Department of Mathematical Modelling, Technical University of Denmark, 1998. QAPpack Home Page: `http://www.imm.dtu.dk/~te/QAPpack/`

[11] A. M. Frieze and J. Yadegar, On the quadratic assignment problem, *Discrete Applied Mathematics* **5**, 1983, 89–98.

[12] P. C. Gilmore, Optimal and suboptimal algorithms for the quadratic assignment problem, *SIAM Journal on Applied Mathematics* **10**, 1962, 305–313.

[13] P. Hahn and T. Grant, Lower bounds for the quadratic assignment problem based upon a dual formulation, to appear in *Operations Research*.

[14] P. Hahn, T. Grant, and N. Hall, Solution of the quadratic assignment problem using the Hungarian method, to appear in *European Journal of Operational Research*.

[15] T. C. Koopmans and M. J. Beckmann, Assignment problems and the location of economic activities, *Econometrica* **25**, 1957, 53–76.

[16] E. L. Lawler, The quadratic assignment problem, *Management Science* **9**, 1963, 586–599.

[17] T. Mautor and C. Roucairol, A new exact algorithm for the solution of quadratic assignment problems, *Discrete Applied Mathematics* **55**, 1994, 281–293.

[18] C. E. Nugent, T. E. Vollmann, and J. Ruml, An experimental comparison of techniques for the assignment of facilities to locations, *Journal of Operations Research* **16**, 1969, 150–173.

[19] P. Pardalos, F. Rendl, and H. Wolkowicz, The quadratic assignment problem: A survey and recent developments, in *Quadratic Assignment and Related Problems*, P. Pardalos and H. Wolkowicz, eds., *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* **16**, 1994, 1–42, AMS, Providence, Rhode Island.

[20] M.G.C. Resende, K.G. Ramakrishnan, and Z. Drezner, Computing lower bounds for the quadratic assignment problem with an interior point algorithm for linear programming, *Operations Research* **43(5)**, 1995, 781–791.

Appendix: Algorithm 3.1 yields GLB, CMB, AJB, AJB' and HGB

Here we illustrate by means of an example that the lower bounds GLB, CMB, AJB, AJB', and HGB as described in Section 2 can be obtained by applying Algorithm 3.1. We proceed as follows: For an example of trivial size ($n = 3$) we compute each of the bounds mentioned above as described in Section 2. After computing a certain bound we apply Algorithm 3.1 with specifications corresponding to that bound and compare the results. The number of iterations performed to compute the bounds is equal to 2, i.e. max_it $= 2$.

Consider a QAP of size $n = 3$ of the form (1) with array of coefficients D and matrix C of coefficients of the linear term given below:

$$D = \left(\begin{array}{ccc|ccc|ccc}
0 & * & * & * & 0 & * & * & * & 0 \\
* & 10 & 7 & 3 & * & 5 & 2 & 1 & * \\
* & 5 & 3 & 4 & * & 2 & 3 & 5 & * \\
\hline
* & 7 & 4 & 1 & * & 9 & 7 & 5 & * \\
0 & * & * & * & 0 & * & * & * & 0 \\
* & 3 & 6 & 7 & * & 6 & 2 & 3 & * \\
\hline
* & 3 & 6 & 3 & * & 5 & 2 & 4 & * \\
* & 5 & 2 & 4 & * & 6 & 5 & 3 & * \\
0 & * & * & * & 0 & * & * & * & 0
\end{array}\right) \qquad C = \begin{pmatrix} 3 & 2 & 3 \\ 1 & 2 & 3 \\ 4 & 5 & 3 \end{pmatrix}$$

The Gilmore-Lawler bound. For each of the 2×2 submatrices of the matrices $D^{(i,j)}$, $1 \le i, j \le 2$, containing only allowed (specified) entries we solve an LAP. Let us denote by z_{ij} the optimal values of these LAPs (see Section 2.2). In general these LAPs could be solved by the Hungarian method, but for $n = 3$ they can be trivially solved as there are only 2 feasible solutions for each of them. For each of these LAPs, we give below an optimal assignment, the optimal value z_{ij}, and optimal values of the dual variables. The latter are going to be used in the other bounding procedures.

$$\begin{array}{r|cc}
z_{11} = 12 & 2 & 0 \\ \hline
7 & 10 & \mathbf{7} \\
3 & \mathbf{5} & 3
\end{array}
\qquad
\begin{array}{r|cc}
z_{12} = 5 & 1 & 0 \\ \hline
2 & \mathbf{3} & 5 \\
2 & 4 & \mathbf{2}
\end{array}
\qquad
\begin{array}{r|cc}
z_{13} = 4 & 1 & 0 \\ \hline
1 & 2 & \mathbf{1} \\
2 & \mathbf{3} & 5
\end{array}
\qquad
\begin{array}{r|cc}
z_{21} = 7 & 0 & 1 \\ \hline
3 & 7 & \mathbf{4} \\
3 & \mathbf{3} & 6
\end{array}
\qquad (17)$$

$$\begin{array}{r|cc} z_{22}=7 & 0 & 5 \\ \hline 1 & \mathbf{1} & 9 \\ 1 & 7 & \mathbf{6} \end{array} \qquad \begin{array}{r|cc} z_{23}=7 & 0 & 1 \\ \hline 4 & 7 & \mathbf{5} \\ 2 & \mathbf{2} & 3 \end{array} \qquad \begin{array}{r|cc} z_{31}=5 & 1 & 0 \\ \hline 2 & \mathbf{3} & 6 \\ 2 & 5 & \mathbf{2} \end{array} \qquad \begin{array}{r|cc} z_{32}=9 & 0 & 2 \\ \hline 3 & \mathbf{3} & 5 \\ 4 & 4 & \mathbf{6} \end{array} \tag{18}$$

$$\begin{array}{r|cc} z_{33}=5 & 1 & 2 \\ \hline 1 & \mathbf{2} & 4 \\ 1 & 5 & \mathbf{3} \end{array} \qquad Z = \begin{pmatrix} 12 & 5 & 4 \\ 7 & 7 & 7 \\ 5 & 9 & 5 \end{pmatrix} \tag{19}$$

Further we solve the leader LAP with cost matrix $C + Z$. Below we present the optimal value and an optimal solution of this LAP, optimal values of its dual variables and the reduced costs.

$$C + Z = \begin{pmatrix} 15 & \mathbf{7} & 7 \\ \mathbf{8} & 9 & 10 \\ 9 & 14 & \mathbf{8} \end{pmatrix} \qquad \begin{array}{r|ccc} GLB = 23 & 1 & 0 & 0 \\ \hline 7 & 7 & 0 & 0 \\ 7 & 0 & 2 & 3 \\ 8 & 0 & 6 & 0 \end{array} \tag{20}$$

Hence the Gilmore-Lawler bound is equal to 23.

It is very easily seen that Algorithm 3.1 with the respective specifications produces GLB= 23. Indeed, after setting all variables but β_{ijkl} and γ_{ij} to 0 and the iteration number to 1 at Step 0, the algorithm performs Step 1b where it solves LAPs with costs $d_{ijkl} \pm \alpha_{ijkl} - \sigma'_{ijk} - \theta'_{ijl} = d_{ijkl}$, $1 \le k, l \le 3$, $k \ne i$, $l \ne j$, for each pair (i, j) with $1 \le i, j \le 3$. But these cost matrices are exactly the 2×2 submatrices of $D^{(i,j)}$ with allowed (specified) entries. The optimal values of the dual variables are then clearly those given above. Further the algorithm performs Step 2 where it solves the leader LAP with costs

$$c_{ij} + \sum_{\substack{k=1 \\ k \ne i}}^{3} (\sigma_{ijk} + \sigma'_{ijk}) + \sum_{\substack{l=1 \\ l \ne j}}^{n} (\theta_{ijl} + \theta'_{ijl}) - \lambda'_i - \mu'_j = c_{ij} + z_{ij}$$

where the last equality holds because of strong duality. The algorithm computes optimal values of the dual variables of the leader LAP, which are the ones presented in the first row and column of the matrix at the right hand side of (20). Again, due to strong duality, the sum of all these dual variables which is output by the algorithm equals the optimal value of the LAP with cost matrix $C + Z$. Hence GLB is indeed obtained by applying Algorithm 3.1.

The bound of Carraresi and Malucelli. For CMB, we compute a reformulation of the QAP according to formulas (12), (13). In the first iteration we start with $\beta_{ijkl} = \sigma_{ijk} = \theta_{ijl} = \gamma_{ij} = 0$ for all $1 \le i, j, k, l \le 3$, and hence let the given QAP unchanged. Then, we compute GLB for the current reformulation, that is for the initial QAP, and update the variables, β_{ijkl}, σ_{ijk}, θ_{ijl} and γ_{ij}. At the end of the first iteration the bound equals GLB=23. Further, σ_{ijk}, $k \ne i$, and θ_{ijl}, $j \ne l$, are set equal to the optimal values of the dual variables of LAP$^{(i,j)}$ solved in the first iteration. Thus, for each (i, j), σ_{ijk} for $k \ne i$, and θ_{ijl} for $l \ne j$, are given by the first column and row of the matrices presented in (17), (18), (19), respectively. Recall that the left-to-right and top-to-bottom order of these matrices corresponds to the lexicographic order of the pairs (i,j). Further we set $\gamma_{ij} := \frac{1}{2}(c_{ij} + z_{ij} - \lambda_i - \mu_j)$ where λ_i and μ_j are optimal values of the dual variables of the LAP with coefficients $C + Z$. Hence

$$(\gamma_{ij}) = \frac{1}{2}\begin{pmatrix} 7 & 0 & 0 \\ 0 & 2 & 3 \\ 0 & 6 & 0 \end{pmatrix}$$

And finally $\beta_{ijkl} = d_{klij} - d_{ijkl}$.

Now by applying (12) and (13) we get the new arrays of coefficients D' and C':

$$D' = \left(\begin{array}{ccc|ccc|ccc} 0 & * & * & * & 0 & * & * & * & 0 \\ * & 1 & 9/2 & 4 & * & 3/2 & 0 & 4 & * \\ * & 3 & 0 & 0 & * & 1 & 4 & 3 & * \\ \hline * & 0 & 0 & 9/2 & * & 0 & 7/2 & 3 & * \\ 0 & * & * & * & 0 & * & * & * & 0 \\ * & 3 & 3 & 2 & * & 0 & 0 & 3 & * \\ \hline * & 1 & 0 & 7/2 & * & 3 & 7/2 & 0 & * \\ * & 7 & 3/2 & 0 & * & 3/2 & 2 & 1 & * \\ 0 & * & * & * & 0 & * & * & * & 0 \end{array}\right) \tag{21}$$

$$C' = C + Z - \begin{pmatrix} 7 & 0 & 0 \\ 0 & 2 & 3 \\ 0 & 6 & 0 \end{pmatrix} = \begin{pmatrix} 8 & 7 & 7 \\ 8 & 7 & 7 \\ 9 & 8 & 8 \end{pmatrix} \tag{22}$$

The new array D' is obtained by substituting d_{ijkl} by the sum of the reduced costs $d_{klij} - \sigma_{kli} - \theta_{klj}$ and γ_{kl}. If we compute GLB for this reformulation we get CMB (recall that the number of iterations was fixed to 2). This is done by solving the block LAPs for the new array D' to obtain the matrix of their optimal values which is given below:

$$Z' = \begin{pmatrix} 1 & 1/2 & 3 \\ 3 & 2 & 3 \\ 5/2 & 3 & 2 \end{pmatrix} \tag{23}$$

Then, we solve the leader LAP with cost matrix $Z' + C'$. The computation of this GLB yields CMB= 28.

What does Algorithm 3.1 do in the case of CMB? We have argued that the first iteration of Algorithm 3.1 produces GLB which in this case equals 23. Then, in Step 3, σ'_{ijk}, θ'_{ijl}, σ_{ijk}, and θ_{ijl}, for $1 \le i,j,k,l \le 3$, $i \ne k$, $j \ne l$ are updated. (Actually the β_{ijkl} are also updated, but in the case of CMB this update is irrelevant because the algorithm does not really use the variables β.) σ'_{ijk} are set equal to the optimal values of the corresponding dual variables of $\mathrm{LAP}^{(i,j)}$ (first column of the submatrix containing z_{ij} in (17), (18), (19)) minus $\frac{1}{2}\gamma_{ij}$. Here γ_{ij} are the reduced costs of the LAP with coefficient matrix $C + Z$, i.e. the entries of the matrix given in the right hand side of (22). θ'_{ijk} are set equal to the optimal values of the corresponding dual variables of $\mathrm{LAP}^{(i,j)}$ (first row of the submatrix containing z_{ij} in (17), (18), (19)). σ_{ijk} and θ_{ijl} are set to 0. Finally a complete interchange of coefficients and variables corresponding to complementary elements is made. Then, the Algorithm goes to Step 1 and solves the block LAPs with costs $d_{ijkl} \pm \alpha_{ijkl} - \sigma'_{ijk} - \theta'_{ijl} = d_{ijkl} - \sigma'_{ijk} - \theta'_{ijl}$ (in the case of CMB, $\alpha_{ijkl} = 0$ in the course of the algorithm). But considering the update of variables described above, the last expression equals the sum of the reduced cost in the position (i,j) of LAP$^{(k,l)}$ and $\frac{1}{2}\gamma_{kl}$. It easy to see that these costs are exactly those presented in (21). In Step 1 the algorithm also computes optimal values for the dual variables σ_{ijk}, θ_{ijl} of the block LAPs. Further, it goes to Step 2 where it solves an LAP with costs

$$c_{ij} + \sum_{\substack{k=1 \\ k\neq i}}^{3} \left(\sigma_{ijk} + \sigma'_{ijk}\right) + \sum_{\substack{l=1 \\ l\neq j}}^{3} \left(\theta_{ijl} + \theta'_{ijl}\right) - \lambda' - \mu'_j =$$

$$c_{ij} + \left(\sum_{\substack{k=1 \\ k\neq i}}^{3} \sigma'_{ijk} + \sum_{\substack{l=1 \\ l\neq j}}^{3} \theta'_{ijl}\right) - \lambda' - \mu'_j + \left(\sum_{\substack{k=1 \\ k\neq i}}^{3} \sigma_{ijk} + \sum_{\substack{l=1 \\ l\neq j}}^{3} \theta_{ijl}\right)$$

The rightmost parenthesis at the right hand side of the above equality is equal to the optimal value of LAP$^{(i,j)}$, i.e. it is equal to z'_{ij}, for all (i,j) (strong duality). Further, $\lambda'_i = \mu'_j = 0$ for all (i,j), and the remaining parenthesis is equal to the optimal value of LAP$^{(i,j)}$ in the previous iteration minus γ_{ij}, i.e. this parenthesis is equal to $z_{ij} - \gamma_{ij}$ for all i,j. Thus, we solve here an LAP with cost matrix $C' + Z'$ as we did when applying the original CMB as described in Section 2.3. Clearly we obtain the same value of 28 for the bound.

The bounds of Adams and Johnson. We now show how AJB and AJB' are obtained by Algorithm 3.1. In each iteration, it solves first the block problems LAP$^{(i,j)}$ with costs $d_{ijkl} - \alpha_{ijkl} - \sigma'_{ijk} - \theta'_{ijl} = d_{ijkl} - \alpha_{ijkl}$ for $i < k$, $j \neq l$, and $d_{ijkl} + \alpha_{ijkl} - \sigma'_{ijk} - \theta'_{ijl} = d_{ijkl} + \alpha_{ijkl}$ for $i > k$, $j \neq l$, for each $1 \leq i,j \leq 3$. The last equalities hold because in the case of AJB (AJB') the algorithm does not change the initial value of σ'_{ijk}, θ'_{ijl} which is equal to 0. The initial values of α_{ijkl} are equal to 0 like in the original algorithm of Adams and Johnson described in Section 2.4. Moreover, in the case of AJB the α-s are updated only in Step 3 by applying the same formula as in Section 2.4. In the case of AJB' the values of α_{ijkl} are updated also in Step 1c. Here we also have the same update as in the original description of AJB' (see Section 2.4). Thus, Algorithm 3.1 solves the same block LAPs as the Adams and Johnson in their original bounds do (see Section 2.4).

Further, in each iteration the algorithm solves a leader LAP with costs

$$c_{ij} + \sum_{\substack{k=1 \\ k\neq i}}^{3} \left(\sigma_{ijk} + \sigma'_{ijk}\right) + \sum_{\substack{l=1 \\ l\neq i}}^{3} \left(\theta_{ijl} + \theta'_{ijk}\right) - \lambda'_i - \mu'_j = c_{ij} + \sum_{\substack{k=1 \\ k\neq i}}^{3} \sigma_{ijk} + \sum_{\substack{l=1 \\ l\neq i}}^{3} \theta_{ijl}$$

The last equality holds because as mentioned above $\sigma'_{ijk} = \theta'_{ijl} = 0$ during the course of the algorithm, and $\lambda'_i = \mu'_j = 0$ as set at the end of each iteration in Step 3. The bounding procedure of Adams and Johnson described in Section 2.4 solves exactly the same LAP. To see this, recall that

$$\sum_{\substack{k=1 \\ k\neq i}}^{n} \sigma_{ijk} + \sum_{\substack{l=1 \\ l\neq i}}^{n} \theta_{ijl}$$

is the optimal value of LAP$^{(i,j)}$.

As an illustration let us perform one iteration of the enhanced procedure of Adams and Johnson to compute AJB' in detail. The parameter setting is the one used by Adams and Johnson in their experiments in [1]: $\varphi_{ijkl} = 1/2$, $\psi_{ij} = 2/3$, for $1 \leq i,j,k,l \leq 3$. In the first iteration, we start with $\alpha_{ijkl} = 0$ for $1 \leq i, j, k, l \leq 3$, $i < k$, $l \neq k$. Solve the first three block LAPs $\mathrm{LAP}^{(1,1)}$, $\mathrm{LAP}^{(1,2)}$, $\mathrm{LAP}^{(1,3)}$ and obtain an optimal solution and the optimal value of these block LAPs, and optimal values of the dual variables as presented in (17). After these steps the updated α_{ijkl} are given as follows:

$$(\alpha_{ijkl}) = \left(\begin{array}{ccc|ccc|ccc}
* & * & * & * & * & * & * & * & * \\
* & 1 & 0 & 0 & * & 3 & 0 & 0 & * \\
* & 0 & 0 & 1 & * & 0 & 0 & 3 & * \\
\hline
* & * & * & * & * & * & * & * & * \\
* & * & * & * & * & * & * & * & * \\
* & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\
\hline
* & * & * & * & * & * & * & * & * \\
* & * & * & * & * & * & * & * & * \\
* & * & * & * & * & * & * & * & *
\end{array}\right) \tag{24}$$

The β-s of the block LAPs of the first row are set to 0. The other β-s remain equal to 0 as set in Step 0 of the algorithm. Thus at this point all β-s are equal to 0. Then, we proceed with $\mathrm{LAP}^{(2,1)}$, $\mathrm{LAP}^{(2,2)}$, $\mathrm{LAP}^{(2,3)}$ with coefficients $d_{ijkl} \pm \alpha_{ijkl}$ and obtain the optimal objective function values and optimal solutions for these ALPs, as well as optimal values of the dual variables:

$$\begin{array}{r|cc}
z_{21} = 7 & 0 & 1 \\
\hline
3 & 7+0 & \mathbf{4+0} \\
3 & \mathbf{3} & 6
\end{array}
\qquad
\begin{array}{r|cc}
z_{22} = 8 & 2 & 6 \\
\hline
0 & \mathbf{1+1} & 9+0 \\
0 & 7 & \mathbf{6}
\end{array}
\qquad
\begin{array}{r|cc}
z_{23} = 10 & 0 & 1 \\
\hline
7 & 7+0 & \mathbf{5+3} \\
2 & \mathbf{2} & 3
\end{array} \tag{25}$$

After these operations, the α-s of the sixth row in (24) are updated as follows:

$$(\alpha_{ijkl}) = \left(\begin{array}{ccc|ccc|ccc}
* & * & * & * & * & * & * & * & * \\
* & 1 & 0 & 0 & * & 3 & 0 & 0 & * \\
* & 0 & 0 & 1 & * & 0 & 0 & 3 & * \\
\hline
* & * & * & * & * & * & * & * & * \\
* & * & * & * & * & * & * & * & * \\
* & 0 & 2 & 5 & * & 0 & 0 & 0 & * \\
\hline
* & * & * & * & * & * & * & * & * \\
* & * & * & * & * & * & * & * & * \\
* & * & * & * & * & * & * & * & *
\end{array}\right) \tag{26}$$

Some of the β-s of the block LAPs in the second row are now different from 0 as presented below:

$$(\beta_{ijkl}) = \left(\begin{array}{ccc|ccc|ccc}
* & * & * & * & * & * & * & * & * \\
* & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\
* & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\
\hline
* & 4 & 0 & 0 & * & 3 & 0 & 0 & * \\
* & * & * & * & * & * & * & * & * \\
* & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\
\hline
* & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\
* & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\
* & * & * & * & * & * & * & * & *
\end{array}\right) \tag{27}$$

Then, problems $\mathrm{LAP}^{(3,1)}$, $\mathrm{LAP}^{(3,2)}$, $\mathrm{LAP}^{(3,3)}$ with coefficients $d_{ijkl} \pm \alpha_{ijkl}$ are solved to obtain the optimal objective function values the optimal values and optimal solutions for these LAPs, as well as optimal values of the dual variables as follows:

$$\begin{array}{r|cc} z_{31}=6 & 4 & 2 \\ \hline 0 & \mathbf{3+1} & 6+0 \\ 0 & 5+5 & \mathbf{2+0} \end{array} \qquad \begin{array}{r|cc} z_{32}=9 & 3 & 6 \\ \hline 0 & \mathbf{3+0} & 5+3 \\ 0 & 4+0 & \mathbf{6+0} \end{array} \qquad \begin{array}{r|cc} z_{33}=5 & 2 & 3 \\ \hline 0 & \mathbf{2+0} & 4+0 \\ 0 & 5+2 & \mathbf{3+0} \end{array} \tag{28}$$

After these operations the α-s do not change whereas the β-s have to be updated as follows:

$$(\beta_{ijkl}) = \left(\begin{array}{ccc|ccc|ccc} * & * & * & * & * & * & * & * & * \\ * & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\ * & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\ \hline * & 4 & 0 & 0 & * & 3 & 0 & 0 & * \\ * & * & * & * & * & * & * & * & * \\ * & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\ \hline * & 0 & 4 & 0 & * & 2 & 0 & 1 & * \\ * & 6 & 0 & 1 & * & 0 & 5 & 0 & * \\ * & * & * & * & * & * & * & * & * \end{array}\right) \tag{29}$$

Further, we must solve the leader LAP with cost matrix $C+Z$ to obtain the lower bound and optimal values of the dual variables, as well as the reduced costs γ_{ij} which are needed in the update formula for the α-s:

$$C+Z = \begin{pmatrix} 15 & \mathbf{7} & 7 \\ \mathbf{8} & 10 & 13 \\ 10 & 14 & \mathbf{8} \end{pmatrix} \qquad \begin{array}{r|ccc} \text{AJB'} = 23 & 1 & 0 & 0 \\ \hline 7 & 7 & 0 & 0 \\ 7 & 0 & 3 & 6 \\ 8 & 1 & 6 & 0 \end{array} \qquad (\gamma_{ij}) = \begin{pmatrix} 7 & 0 & 0 \\ 0 & 3 & 6 \\ 1 & 6 & 0 \end{pmatrix} \tag{30}$$

Hence, AJB' equals 23 after the first iteration. In the second iteration we proceed in the same way. The only difference consists in the values of the α-s, which are updated as follows for $1 \le i,j,k,l \le 3$, $i < k$, $j \ne l$ by

$$\alpha_{ijkl} = \alpha_{ijkl} + 0.5(\beta_{ijkl} + \frac{1}{3}\gamma_{ij}) - 0.5(\beta_{klij} + \frac{1}{3}\gamma_{kl})$$

yielding

$$(\alpha_{ijkl}) = \left(\begin{array}{ccc|ccc|ccc} * & * & * & * & * & * & * & * & * \\ * & 2/3 & 1/6 & -2 & * & -1 & 0 & -2 & * \\ * & 1/6 & 7/6 & -1/6 & * & -1/2 & -13/6 & -2 & * \\ \hline * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & -3/2 & -5/2 & -8/3 & * & 1/2 & 5/6 & 0 & * \\ \hline * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \\ * & * & * & * & * & * & * & * & * \end{array}\right)$$

Performing the second iteration using the updated α-s yields the bound AJB'=$29\frac{1}{3}$.

The bound of Hahn and Grant. Let us compute HGB for our example by applying two iterations of the procedure of Hahn and Grant described in Section 2.5. First we collect the complementary elements in the uppermost position in array D. Then, the transformed array D' looks as follows:

$$D' = \left(\begin{array}{ccc|ccc|ccc} 0 & * & * & * & 0 & * & * & * & 0 \\ * & 11 & 14 & 10 & * & 10 & 6 & 10 & * \\ * & 8 & 5 & 7 & * & 6 & 9 & 10 & * \\ \hline * & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\ 0 & * & * & * & 0 & * & * & * & 0 \\ * & 7 & 11 & 12 & * & 9 & 4 & 9 & * \\ \hline * & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\ * & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\ 0 & * & * & * & 0 & * & * & * & 0 \end{array}\right) \tag{31}$$

In the case of Algorithm 3.1 the collection of complementary costs is done by setting $\alpha_{ijkl} = \alpha_{ijkl} - \beta_{klij}$ for $i < k$, $l \neq j$ and $\alpha_{klij} = \alpha_{klij} + \beta_{klij}$ for $i > k$, $l \neq j$, for any pair (i,j) (see Step 1a). In the first iteration we have then $\alpha_{ijkl} = -\beta_{klij} = -d_{klij}$. The algorithm collects the complementary costs corresponding to a block LAP at the moment where this LAP is going to be solved (Step 1b), whereas in the original bounding procedure of Hahn and Grant this is done row-wise in terms of rows of block LAPs. Next we solve the block LAPs of the first row for array D'. In the case of Algorithm 3.1 this is done in Step 1b, where we solve LAP$^{(1,j)}$ with coefficients

$$d_{1jkl} - \alpha_{1jkl} - \sigma'_{1jk} - \theta'_{1jl} \quad \text{if } 1 < k \tag{32}$$

which equals $d_{1jkl} + d_{kl1j}$ for $1 < k$, $l \neq j$.

After solving the first row of block LAPs and adding their optimal values to the corresponding leaders, the transformed array D' looks as follows:

$$D' = \left(\begin{array}{ccc|ccc|ccc} 16 & * & * & * & 16 & * & * & * & 16 \\ * & 0 & 6 & 0 & * & 1 & 0 & 0 & * \\ * & 0 & 0 & 0 & * & 0 & 3 & 0 & * \\ \hline * & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\ 0 & * & * & * & 0 & * & * & * & 0 \\ * & 7 & 11 & 12 & * & 9 & 4 & 9 & * \\ \hline * & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\ * & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\ 0 & * & * & * & 0 & * & * & * & 0 \end{array}\right) \tag{33}$$

Next we proceed with the second block-row by adding complementary costs (positive slacks from the first block-row) and solving LAP$^{(2,1)}$, LAP$^{(2,2)}$ and LAP$^{(2,3)}$:

$$\begin{array}{r|cc} z_{21} = 7 & 0 & 0 \\ \hline 0 & 0+0 & \mathbf{0+0} \\ 7 & \mathbf{7} & 11 \end{array} \qquad \begin{array}{r|cc} z_{22} = 9 & 0 & 0 \\ \hline 0 & \mathbf{0+0} & 0+0 \\ 9 & 12 & \mathbf{9} \end{array} \qquad \begin{array}{r|cc} z_{23} = 5 & 4 & 1 \\ \hline 0 & 0+6 & \mathbf{0+1} \\ 0 & \mathbf{4} & 9 \end{array}$$

How does Algorithm 3.1 obtain the LAPs of the second block-row? After processing the first block-row the values of the α-s are given by $\alpha_{1jkl} = -d_{kl1j}$ for $i < k$, $l \neq j$ and $\alpha_{ijkl} = 0$ for $1 < i < k$, $l \neq j$. At this moment the values of β-s are as follows: for each j β_{1jkl}, $1 < k$, $l \neq j$, are equal to the reduced costs of the LAP$^{(1,j)}$, $\beta_{kl1j} = 0$, for all $k > i$, $l \neq j$ and for the remaining indices (i,j,k,l), $i \neq 1$, $k \neq 1$, $\beta_{ijkl} = d_{ijkl}$. Let us see how Algorithm 3.1

produces LAP$^{(2,1)}$. Everything is analogous for the other block LAPs of the second block-row. In Step 1a we have

$$\alpha_{213l} = \alpha_{213l} - \beta_{3l21} = -d_{3l21}, \quad \beta_{3l21} = 0 \quad \text{for all} \quad l \neq 1$$
$$\alpha_{1l21} = \alpha_{1l21} + \beta_{1l21} = -d_{211l} + \beta_{1l21}, \quad \beta_{1l21} = 0 \quad \text{for all} \quad l \neq 1$$

Then LAP$^{(2,1)}$ with coefficients given as below is solved:

$$d_{211l} + \alpha_{1l21} - \sigma'_{211} - \theta'_{21l} = \beta_{1l21} \quad \text{for} \quad l \neq 1 \tag{34}$$
$$d_{213l} - \alpha_{213l} - \sigma'_{213} - \theta_{21l} = d_{213l} + d_{3l21} \quad \text{for} \quad l \neq 1 \tag{35}$$

Now, considering that β_{1l21} are the reduced costs of LAP$^{(1,l)}$, we indeed obtain LAP$^{(2,1)}$ with the same coefficient as when applying the original bounding procedure of Hahn and Grant.

Finally, we move slacks to the third block-row, and process LAP$^{(3,1)}$, LAP$^{(3,2)}$ and LAP$^{(3,3)}$:

$z_{31}=0$	0	0
0	$\mathbf{0+0}$	$0+3$
0	$0+3$	$\mathbf{0+0}$

$z_{32}=0$	0	0
0	$0+0$	$\mathbf{0+0}$
0	$\mathbf{0+0}$	$0+8$

$z_{33}=0$	0	0
0	$\mathbf{0+0}$	$0+0$
0	$0+4$	$\mathbf{0+0}$

Analogously as for the block LAPs of the second block-row, one can check that Algorithm 3.1 solves the same problems LAP$^{(3,j)}$, $1 \leq j \leq 3$ in Step 1b.

Similarly as for other bounds let us denote the matrix of the leaders by $Z = (z_{ij})$. We add Z to the cost matrix C and solve a leader LAP with cost $Z + C$. In the case of Algorithm 3.1 this is done in Step 2, where we solve an LAP with costs

$$c_{ij} + \left(\sum_{\substack{k=1 \\ k \neq i}}^{n} \sigma'_{ijk} + \sum_{\substack{l=1 \\ l \neq j}}^{n} \theta'_{ijl} \right) - \lambda'_i - \mu'_j + \left(\sum_{\substack{k=1 \\ k \neq i}}^{n} \sigma_{ijk} + \sum_{\substack{l=1 \\ l \neq j}}^{n} \theta_{ijl} \right) \tag{36}$$

This is equal to $c_{ij} + z_{ij}$ in the first iteration, because $\sigma'_{ijk} = \theta'_{ijl} = \lambda'_i = \mu'_j = 0$ and $\sum_k \sigma_{ijk} + \sum_l \theta_{ijl} = z_{ij}$ due to strong duality. The matrix $C + Z$, the optimal value of the corresponding leader LAP together with the reduced costs and optimal values of the dual variables are represented below.

$$C + Z = \begin{pmatrix} 19 & 18 & 19 \\ 8 & 11 & 8 \\ 4 & 5 & 3 \end{pmatrix} \qquad \begin{array}{r|ccc} \text{HGB} = 29 & 1 & 2 & 0 \\ \hline 16 & 2 & 0 & 3 \\ 7 & 0 & 2 & 1 \\ 3 & 0 & 0 & 0 \end{array} \qquad \frac{1}{2}(\gamma_{ij}) = \begin{pmatrix} 1 & 0 & 3/2 \\ 0 & 1 & 1/2 \\ 0 & 0 & 0 \end{pmatrix} \tag{37}$$

Then we redistribute the non-zero reduced costs of the leader LAP to the corresponding block LAPs and recollect the complementary elements in the upper most position of the array D'. Let us still denote the transformed array by D'. D' is given as follows:

$$D' = \left(\begin{array}{ccc|ccc|ccc} 0 & * & * & * & 0 & * & * & * & 0 \\ * & 2 & 7/2 & 0 & * & 1/2 & 3/2 & 5/2 & * \\ * & 1 & 1 & 0 & * & 0 & 9/2 & 3/2 & * \\ \hline * & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\ 0 & * & * & * & 0 & * & * & * & 0 \\ * & 0 & 4 & 4 & * & 0 & 1/2 & 17/2 & * \\ \hline * & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\ * & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\ 0 & * & * & * & 0 & * & * & * & 0 \end{array}\right) \qquad (38)$$

At this point Algorithm 3.1 performs Step 3, where the variables σ'_{ijk} and θ'_{ijl} are updated. The latter contain the values of the dual variables for the block LAPs. The σ'_{ijk} are obtained by subtracting a half of the reduced cost, $\frac{1}{2}\gamma_{ij}$, from the optimal values of the dual variables of $\mathrm{LAP}^{(i,j)}$. β_{ijkl} are also updated by adding to them $\frac{1}{2}\gamma_{ij}$. It is not difficult to check that before this update $\beta_{ijkl} = 0$ if $i < k$, $j \neq l$ and the other variables β_{ijkl}, $i > k$, $j \neq l$, equal the reduced costs of the corresponding problems $\mathrm{LAP}^{(i,j)}$. Let us denote these reduced costs by $\bar{\beta}_{ijkl}$. As for the α-s it can be checked that at the end of the fisrt iteration of Algorithm 3.1 they are given as $\alpha_{ijkl} = -d_{klij} + \bar{\beta}_{ijkl}$, for all $i < k$, $l \neq j$. Next the algorithm performs Step 1a for the LAPs of the first block-row. We show that $\mathrm{LAP}^{(1,1)}$ solved by Algorithm 3.1 has the same coefficients as $\mathrm{LAP}^{(1,1)}$ solved by the original bounding procedure of Hahn and Grant. (The coefficients of this LAP are the entries in the left-upper corner of array D'.) Analogously, we could show the same for the other LAPs of the first block-row. By applying Step 1a we get:

$$\alpha_{11kl} = \alpha_{11kl} - \beta_{kl11} = -d_{kl11} + \bar{\beta}_{11kl} - \bar{\beta}_{kl11} - \frac{1}{2}\gamma_{kl}\,, \qquad \text{for } k > 1,\, l \neq 1\,.$$

Now the coefficients of $\mathrm{LAP}^{(1,1)}$ solved in Step 1b are given as in (32). By considering the updates described above, we get that

$$d_{11kl} + d_{kl11} - \bar{\beta}_{11kl} + \bar{\beta}_{kl11} + \frac{1}{2}\gamma_{kl} - \sigma_{1jk} - \theta_{1jl} + \frac{1}{2}\gamma_{11} = \frac{1}{2}\gamma_{11} + \bar{\beta}_{kl11} + \frac{1}{2}\gamma_{kl} \qquad (39)$$

for $k > 1$, $l \neq 1$. Now $\frac{1}{2}\gamma_{11}$ is the entry in position $(1,1,k,l)$, $k > 1$, $l > 1$, and analogously, $\bar{\beta}_{kl11} + \frac{1}{2}\gamma_{kl}$ is in position $(k,l,1,1)$, after the redistribution of the leaders. This shows what we claimed.

Now, let us take a closer look at the coefficients of the $\mathrm{LAP}^{(2,1)}$, $\mathrm{LAP}^{(2,2)}$, $\mathrm{LAP}^{(2,3)}$ solved by Algorithm 3.1 in Step 1b. Consider for instance $\mathrm{LAP}^{(2,1)}$ (the discussion for the other LAPs would be analogous). After having processed the LAPs of the first block-row the α-s and the β-s are given as follows:

$$\begin{aligned} \alpha_{1jkl} &= -d_{kl1j} + \bar{\beta}_{1jkl} - \bar{\beta}_{kl1j} - \tfrac{1}{2}\gamma_{kl}\,, && \text{for } k > 1,\, l \neq j\,, \\ \alpha_{2j3l} &= -d_{kl2j} + \bar{\beta}_{2jkl}\,, && \text{for } l \neq j \\ \beta_{1jkl} &= \text{reduced costs of } \mathrm{LAP}^{(1,j)} = \tilde{\beta}_{1jkl}\,, && \text{for } k > 1,\, l \neq j \\ \beta_{kl1j} &= 0\,, && \text{for all } k > 1,\, l \neq j\,, \end{aligned}$$

and the other β-s, i.e. β_{ijkl} for $i \neq 1$, $k \neq 1$ have the same values they had at the end of the first iteration. Next, in Steps 1a we get

$$\begin{array}{rll} \alpha_{1l21} = & \alpha_{1l21} + \beta_{1l21} = -d_{211l} + \bar{\beta}_{1l21} - \bar{\beta}_{211l} - \frac{1}{2}\gamma_{21} + \tilde{\beta}_{1l21}\,, & \text{for } l \neq 1 \\ \alpha_{213l} = & \alpha_{213l} - \beta_{3l21} = -d_{3l21} + \bar{\beta}_{213l} - \bar{\beta}_{3l21} - \frac{1}{2}\gamma_{3l}\,, & \text{for } l \neq 1 \end{array}$$

Then in Step 1b we solve $\mathrm{LAP}^{(2,1)}$ with coefficients given by

$$d_{211j} + \alpha_{1j21} - \sigma'_{211} - \theta'_{21j} + \frac{1}{2}\gamma_{21} = \tilde{\beta}_{1j21} \qquad \text{for } j \neq 1 \tag{40}$$

$$d_{213j} - \alpha_{213j} - \sigma'_{213} - \theta'_{21j} + \frac{1}{2}\gamma_{21} = d_{213j} + d_{3j21} - \bar{\beta}_{213j} + \bar{\beta}_{3j21} +$$

$$\frac{1}{2}\gamma_{3j} - \sigma'_{213} - \theta'_{21j} + \frac{1}{2}\gamma_{21} = \frac{1}{2}\gamma_{21} + \bar{\beta}_{3j21} + \frac{1}{2}\gamma_{3j}\,, \qquad \text{for } l \neq 1 \tag{41}$$

Equality (40) holds because $\bar{\beta}_{1j21} - \bar{\beta}_{211j} - \sigma'_{211} - \theta'_{21j} = 0$ due to the fact that $\bar{\beta}_{1j21}$ is the coest coefficient at position $(1, j)$ of $\mathrm{LAP}^{(2,1)}$ at the first iteration. Equality (41) is similar to equality (39) and holds for similar reasons. Now, the expressions in the right hand sides of (40) and (41) are exactly the costs of $\mathrm{LAP}^{(2,1)}$ in the procedure of Hahn and Grant, obtained by adding complementary to costs to the second block-row of matrix D', and this proves what we claimed.

Next the third block-row is processed by solving the block LAPs after having collected complementary costs there. Similarly as for the second block-row one can show that the problems $\mathrm{LAP}^{(3,1)}$, $\mathrm{LAP}^{(3,2)}$, $\mathrm{LAP}^{(3,3)}$ solved by Algorithm 3.1 are the same as those solved when applying the original bounding procedure of Hahn and Grant. After having solved all block LAPs we obtain the transformed array D' where the leaders contain the optimal values z'_{ij} of $\mathrm{LAP}^{(i,j)}$, for $1 \leq i, j \leq 3$:

$$D' = \left(\begin{array}{ccc|ccc|ccc} 3 & * & * & * & 0 & * & * & * & 3 \\ * & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\ * & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\ \hline * & 0 & 0 & 0 & * & 0 & 1 & 0 & * \\ 0 & * & * & * & 1 & * & * & * & 1 \\ * & 0 & 0 & 0 & * & 0 & 0 & 0 & * \\ \hline * & 0 & 3 & 0 & * & 0 & 0 & 0 & * \\ * & 4 & 0 & 0 & * & 8 & 4 & 0 & * \\ 0 & * & * & * & 0 & * & * & * & 0 \end{array}\right) \tag{42}$$

Further we solve a leader LAP with coefficients given by the new matrix of the leaders $Z' = (z'_{ij})$ and update the current HGB by adding to it the optimal value of this last LAP, which in our case equal 0. Thus the second iteration does not improve HGB and HGB=29. Algorithm 3.1 solves the leader LAP in Step 2. This LAP has costs given as in (36). Here, for each pair (i, j), σ_{ijk} and θ_{ijl} are the optimal values of the dual variables of $\mathrm{LAP}^{(i,j)}$ solved in the second iteration. Hence $\sum_k \sigma_{ijk} + \sum_l \theta_{ijl} = z'_{ij}$ due to strong duality. Recall that θ'_{ijl} are the optimal values of the dual variables for the $\mathrm{LAP}^{(i,j)}$ solved in the previous iteration. The values of σ'_{ijk} are obtained by subtracting $\frac{1}{2}\gamma_{ij}$ from the optimal values of the dual variables of the same block LAP. This implies that

$$c_{ij} + \sum_k \sigma'_{ijk} + \sum_l \theta'_{ijl} = c_{ij} + z_{ij} - \frac{1}{2}\gamma_{ij}\,,$$

for all $1 \leq i, j \leq 3$. Finally, λ'_i and μ'_j are equal to 0 (as set in Step 3 in the first iteration). Combining these facts implies that the cost coefficients of the leader LAP solved in the second iteration are $c_{ij} + z_{ij} - \frac{1}{2}\gamma_{ij} + z'_{ij}$, for all $1 \leq i, j \leq 3$. But the matrix with costs $c_{ij} + z_{ij} - \gamma_{ij}$ is a sum matrix with entries given as sums $\lambda_i + \mu_j$, where λ_i, μ_j are the optimal values of the dual variables of the leader LAP solved at the end of the first iteration. It is well known and easily seen that an LAP whose coefficient matrix is a sum matrix, is a constant problem, in the sense that every feasible solution yields the same value of the objective function. It is easy to see that in our case this constant equals $\sum_i \lambda_i + \sum_j \mu_j$. But this sum is equal to the HGB obtained at the end of the first iteration (due to strong duality). Further, from the fact that the LAP with costs $c_{ij} + z_{ij} - \gamma_{ij}$ is a constant problem, it follows that the optimal value of the LAP with costs $c_{ij} + z_{ij} - \gamma_{ij} + z'_{ij}$ equals the optimal value of the LAP with costs z'_{ij}, which is 0 in our case, plus the optimal value of the constant LAP which – as argued above – is equal to HGB. This completes our illustrative argument that Algorithm 3.1 indeed produces HGB=29.