Zuse Institute Berlin

Takustr. 7
14195 Berlin
Germany

KOICHI FUJII, NAOKI ITO, SUNYOUNG KIM, MASAKAZU KOJIMA, YUJI SHINANO, KIM-CHUAN TOH

Solving Challenging Large Scale QAPs

We report our progress on the project for solving larger scale quadratic assignment problems (QAPs). Our main approach to solve large scale NP-hard combinatorial optimization problems such as QAPs is a parallel branch-and-bound method efficiently implemented on a powerful computer system using the Ubiquity Generator (UG) framework that can utilize more than 100,000 cores. Lower bounding procedures incorporated in the branch-and-bound method play a crucial role in solving the problems. For a strong lower bounding procedure, we employ the Lagrangian doubly nonnegative (DNN) relaxation and the Newton-bracketing method developed by the authors' group. In this report, we describe some basic tools used in the project including the lower bounding procedure and branching rules, and present some preliminary numerical results.
Our next target problem is QAPs with dimension at least 50, as we have succeeded to solve tai30a and sko42 from QAPLIB for the first time.

ZIB Report 21-02 (January 2021)

Zuse Institute Berlin
Takustr. 7
14195 Berlin
Germany

Telephone: +49 30-84185-0
Telefax: +49 30-84185-125

E-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Solving Challenging Large Scale QAPs

Koichi Fujii[*], Naoki Ito[†], Sunyoung Kim[‡], Masakazu Kojima[§], Yuji Shinano[¶], Kim-Chuan Toh[‖]

January 23, 2021

Abstract

We report our progress on the project for solving larger scale quadratic assignment problems (QAPs). Our main approach to solve large scale NP-hard combinatorial optimization problems such as QAPs is a parallel branch-and-bound method efficiently implemented on a powerful computer system using the Ubiquity Generator (UG) framework that can utilize more than 100,000 cores. Lower bounding procedures incorporated in the branch-and-bound method play a crucial role in solving the problems. For a strong lower bounding procedure, we employ the Lagrangian doubly nonnegative (DNN) relaxation and the Newton-bracketing method developed by the authors' group. In this report, we describe some basic tools used in the project including the lower bounding procedure and branching rules, and present some preliminary numerical results.

Our next target problem is QAPs with dimension at least 50, as we have succeeded to solve tai30a and sko42 from QAPLIB for the first time.

1 Introduction

For a positive integer n, we let $N = \{1, \ldots, n\}$ represent a set of locations and also a set of facilities. Given $n \times n$ symmetric matrices $\boldsymbol{A} = [a_{ik}]$, $\boldsymbol{B} = [b_{j\ell}]$ and an $n \times n$ matrix $\boldsymbol{C} = [c_{ij}]$, the quadratic assignment problem (QAP) is stated as

$$\min_{\pi} \sum_{i \in N} \sum_{k \in N} a_{ik} b_{\pi(i),\pi(k)} + \sum_{i \in N} c_{i,\pi(i)}, \tag{1}$$

where a_{ik} denotes the flow between facilities i and k, $b_{j\ell}$ is the distance between locations j and ℓ, c_{ij} the fixed cost of assigning facility i to location j, and $(\pi(1), \ldots, \pi(n))$ a permutation of $1, \ldots, n$ such that $\pi(i) = j$ if facility i is assigned to location j.

The QAP is known as NP-hard in theory, and solving QAP instances of size larger than 35 in practice still remains challenging. Various heuristic methods for the QAP such as tabu search [22, 23], genetic method [5] and simulated annealing [4] have been developed. Those methods frequently attain near-optimal solutions, which often also happen to be the exact optimal solutions. The exactness is, however, not guaranteed in general.

Most existing methods for finding the exact solutions of QAPs are designed with the branch-and-bound (B&B) method [1, 3, 6, 8, 14, 16]. As its name indicates, branching and (lower) bounding are the main procedures of the method. The bounding based on doubly nonnegative (DNN) relaxations has recently attracted a great deal of attention as it provides tight bounds. Among the methods for solving large scale DNN relaxation problems, we mention four types of methods that can be applied to such DNN problems.

First two of the methods are SDPNAL+ [25] (a majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints) and BBCPOP [9] (a bisection

[*]NTT DATA Mathematical Systems Inc., Tokyo, 160-0016 Japan
[†]Fast Retailing Co., Ltd., Uniqlo City Tokyo, Tokyo, 135-0063 Japan
[‡]Department of Mathematics, Ewha W. University, 52 Ewhayeodae-gil, Sudaemoon-gu, Seoul 120-750, Korea
[§]Department of Industrial and Systems Engineering, Chuo University, Tokyo 192-0393, Japan
[¶]Department of Applied Algorithmic Intelligence Methods (A^2IM), Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany
[‖]Department of Mathematics, and Institute of Operations Research and Analytics, National University of Singapore, 10 Lower Kent Ridge Road, Singapore 119076

and projection method for Binary, Box and Complementary constrained Polynomial Optimization Problems). Some numerical results on these two methods applied to QAP instances with dimensions $n = 15$ to 50 from QAPLIB [7] were reported in [9]. where BBCPOP attained tighter lower bounds for many of instances with dimensions 30 to 50 in less execution time. In addition, new and improved lower bounds were computed by Mittelmann using BBCPOP for the unknown minimum values of larger scale QAP instances including tai100a and tai100b. See QAPLIB [7]. The third method is an alternating direction method of multipliers (ADMM) proposed by Oliveira et al. [13] in combination with the facial reduction for robustness. The fourth method is the Newton-bracketing (NB) method [2, 10], which was developed to further improve the lower bounds obtained from BBCPOP by incorporating the Newton method into BBCPOP.

Numerical results on the NB method and BBCPOP are given in Table 2 of [10], and ones on ADMM in Table 7 of [13]. Lower bounds for the QAP instances sko42, sko49, sko56, sko64, sko72, sko81, tai60a, and tai80a were commonly computed by these methods. While the lower bounds obtained by the NB method for the first three instances are not smaller than those by ADMM and the difference is as big as 1, the NB method clearly attained the tightest lower bound among the three methods for the last five larger-scale instances with dimension $n \geq 60$.

RLT [17] is known as a powerful method for global optimization of polynomial programming problems. In their paper [6], Goncalves et al. implemented the level 2 RLT in their B&B method run on heterogeneous CPUs and GPUs environment. They solved tai35b and tai40b from QAPLIB for the first time.

The main motivation of our project is to challenge larger scale QAP instances from QAPLIB [7] that have not been solved yet. We implement the NB method [10] combined with the B&B method in the specialized Ubiquity Generator (UG) framework [20] to find the exact solutions of large scale QAP instances.

The NB method has the following advantages:

(a) The NB method generates a sequence of intervals $[\mathrm{lb}^k, \mathrm{ub}^k]$ $(k = 1, 2, \ldots)$ such that each $[\mathrm{lb}^k, \mathrm{ub}^k]$ contains the lower bound $\underline{\zeta}$ to be computed and both lb^k and ub^k converge monotonically to $\underline{\zeta}$ as $k \to \infty$. Hence, if lb^k becomes larger than the incumbent objective value $\hat{\zeta}$ at some iteration k, then $\hat{\zeta} < \mathrm{lb}^k \leq \underline{\zeta}$ follows. Consequently, the node can be pruned. If $\mathrm{ub}^k < \hat{\zeta}$ occurs at some iteration k, then branching the node should be determined as $\underline{\zeta} \leq \mathrm{ub}^k < \hat{\zeta}$ holds. In these two cases, the NB method can terminate at iteration k before the interval attains an accurate lower bound. This feature of the NB method works effectively to reduce a considerable amount of the computational time.

(b) The lower bounds computed by the NB method (and also the lower bounds computed by BBCPOP) are reliable or valid in the sense that they are guaranteed by very accurate dual feasible solutions of the Lagrangian DNN relaxation problem (see [10, Section 2.3].) It would be plausible to incorporate the technique used there for ADMM's lower bound, but the resulting lower bound would deteriorate.

The UG is a generic framework to parallelize branch-and-bound based solvers and has achieved large-scale MPI parallelism with 80,000 cores [18]. It has also been generalized to handle non-branch-and-bound based solvers. Its experimental implementation for solving Shortest Vector Problem (SVP) handles more than 100,000 cores [24] .

The aim of this article is to present fundamentals of our joint project for solving larger scale QAPs, report the progress, and discuss future plans to further develop the methods. We mention that tai35b, tai40b, tai30a and sko42 from QAPLIB [7] have been successfully solved by our method, where the last two instances could be solved for the first time. QAPs with dimension at least 50 are our next target problems.

In Section 2, we first reformulate QAP (1) to QAP (2), which is a binary quadratic optimization problem form in $(1+n^2)$-dimensional vector variable $\boldsymbol{u}$. Then we introduce a class of its sub QAPs and describe the enumeration tree of those subproblems used in the branching process. In Section 3, a simple Lagrangian DNN relaxation problem of a sub QAP and the NB method to solve the relaxation problem are presented for the lower bounding procedure. In Section 4, a method to retrieve a feasible solution of (2) from an approximate optimal solution of the Lagrangian DNN relaxation problem is described for the upper bounding procedure. We present three types branching rules in Section 5, and preliminary numerical results in Section 6. Some additional techniques are proposed for future development in Section 7.

Notation and Symbols

Let $\mathbb{R}^m$ denote the m-dimensional Euclidean space of column vectors $\boldsymbol{z} = [z_1; \ldots; z_m]$, $\mathbb{R}^m_+$ the nonnegative orthant of $\mathbb{R}^m$, $\mathbb{S}^m$ the linear space of $m \times m$ symmetric matrices with the inner product $\langle \boldsymbol{A}, \boldsymbol{B} \rangle =$

$\sum_{i=1}^m \sum_{j=1}^m A_{ij}B_{ij}$, and $\mathbb{S}^m_+$ the cone of positive semidefinite matrices in $\mathbb{S}^m$. For every $\ell \times m$ matrix $\boldsymbol{U}$, $\boldsymbol{U}_{\cdot j}$ denotes the jth column vector of $\boldsymbol{U}$, and vec $\boldsymbol{U}$ the ℓm-dimensional column vector converted from the matrix $\boldsymbol{U}$, *i.e.*, $\mathbf{vec}\boldsymbol{U} = [\boldsymbol{U}_{\cdot 1}; \ldots; \boldsymbol{U}_{\cdot m}]$. $\boldsymbol{U}^T$ denotes the transposition of $\boldsymbol{U}$. Throughout the paper, we use n for the size of QAP (1), and $N = \{1, \ldots, n\}$ for the set of facilities and the set of locations. When $\mathbb{R}^{\{0,N\times N\}}$ is used, each $(1+n^2)$-dimensional column vector $\boldsymbol{u} \in \mathbb{R}^{\{0,N\times N\}}$ is represented component-wisely as $[u_0; u_{11}, \ldots; u_{n1}; u_{12}; \ldots; u_{n2}; \ldots, u_{1n}; \ldots, u_{nn}]$, where $\boldsymbol{U} = [u_{ik}]$ forms an $n \times n$ matrix and $\boldsymbol{u} = [u_0; \mathrm{vec}\ \boldsymbol{U}] = [u_0; \boldsymbol{U}_{\cdot 1}; \ldots; \boldsymbol{U}_{\cdot n}]$. Each $((1+n^2) \times (1+n^2))$-dimensional symmetric matrix $\boldsymbol{X} \in \mathbb{S}^{\{0,N\times N\}}$ has elements $X_{\boldsymbol{\alpha}\boldsymbol{\beta}}$ $(\boldsymbol{\alpha}, \boldsymbol{\beta} \in \{0, N \times N\})$.

2 A class of sub QAPs and the enumeration tree

2.1 Conversion of QAP (1) into a QOP with a binary vector variable

Let $\boldsymbol{U} = [u_{ij}]$ be an $n \times n$ matrix variable and $\boldsymbol{u} = [u_0; \mathbf{vec}(\boldsymbol{U})] = [u_0; \boldsymbol{U}_{\cdot 1}; \ldots; \boldsymbol{U}_{\cdot n}] \in \mathbb{R}^{\{0,N\times N\}}$ a $(1+n^2)$-dimensional column vector variable, where

$$u_{ij} = \begin{cases} 1 & \text{if facility } i \text{ is assigned to location } j, \\ 0 & \text{otherwise.} \end{cases}$$

Define a constant matrix

$$\boldsymbol{Q}^0 \quad = \quad \begin{pmatrix} 0 & \mathbf{vec}(\boldsymbol{C})^T/2 \\ \mathbf{vec}(\boldsymbol{C})/2 & \boldsymbol{B} \otimes \boldsymbol{A} \end{pmatrix} \in \mathbb{S}^{\{0,N\times N\}},$$

where $\otimes$ denotes the Kronecker product. Then, the QAP can be expressed as

$$\zeta^0 \quad = \quad \min \left\{ \langle \boldsymbol{Q}^0, \boldsymbol{u}\boldsymbol{u}^T \rangle : \begin{array}{l} \boldsymbol{u} \in \mathbb{R}^{\{0,N\times N\}}_+,\ u_0 = 1, \\ u_{ij}(u_0 - u_{ij}) = 0\ ((i,j) \in N \times N), \\ \displaystyle\sum_{k\in N} u_{ik} = u_0\ (i \in N),\ \sum_{k\in N} u_{kj} = u_0\ (j \in N) \end{array} \right\}. \tag{2}$$

2.2 A class of subproblems of QAP (2)

We describe subproblems of QAP (2) with a family of subsequences $(i_1, \ldots, i_r)$ with $1 \le i_1 < \cdots < i_r \le n$. More precisely, let Π_r denote the family of subsequences of r distinct elements of N $(r = 1, \ldots, n)$. We assume $\Pi_0 = \{\emptyset\}$. Π_n corresponds to the family of permutations of $1, \ldots, n$, and each $F = (i_1, \ldots, i_r) \in \Pi_r$ (or $L = (j_1, \ldots, j_r) \in \Pi_r$) with $r \ge 1$ corresponds to a permutation of r distinct elements i_p $(p = 1, \ldots, r)$ (or j_p $(p = 1, \ldots, r)$) of N. For simplicity of notation, $F \in \Pi_r$ is frequently regarded as a subset of N when $i \in F$ and its complement $F^c = N \backslash F$ are described with respect to N.

For each $(F, L) = ((i_1, \ldots, i_r), (j_1, \ldots, j_r)) \in \Pi_r \times \Pi_r$ $(r \in \{0, 1, \ldots, n\})$, let

$$\begin{array}{rcl} \mathbb{R}^{\{0,F^c\times L^c\}} & = & 1 + |F^c||L^c|\text{-dimensional Euclidean space of column vectors } \boldsymbol{x} \\ & & \text{with elements } x_0 \text{ and } x_{ij}\ ((i,j) \in F^c \times L^c), \\ \mathbb{R}^{\{0,F^c\times L^c\}}_+ & = & \text{the nonnegative orthant of } \mathbb{R}^{\{0,F^c\times L^c\}}, \\ \mathbb{S}^{\{0,F^c\times L^c\}} & = & \text{the linear space of } (1+|F^c||L^c|) \times (1+|F^c||L^c|) \text{ symmetric} \\ & & \text{matrices } \boldsymbol{X} \text{ with elements } X_{\boldsymbol{\alpha}\boldsymbol{\beta}}\ (\boldsymbol{\alpha}, \boldsymbol{\beta} \in \{0, F^c \times L^c\}), \\ \mathbb{S}^{\{0,F^c\times L^c\}}_+ & = & \text{the cone of positive semidefinite matrices in } \mathbb{S}^{\{0,F^c\times L^c\}}. \end{array}$$

Then, for each $(F, L) = ((i_1, \ldots, i_r), (j_1, \ldots, j_r)) \in \Pi_r \times \Pi_r$ $(r \in \{0, 1, \ldots, n\})$, a subproblem of QAP (2)

for assigning each facility $i_p \in F$ to each location $j_p \in L$ $(p = 1, \ldots, r)$ can be written as

$$\mathrm{QAP}(F,L)\colon\ \zeta(F,L) = \min \left\{ \langle \boldsymbol{Q}(F,L),\ \boldsymbol{x}\boldsymbol{x}^T \rangle : \begin{array}{l} \boldsymbol{x} \in \mathbb{R}_+^{\{0,F^c \times L^c\}},\ x_0 = 1, \\ x_{ij}(x_0 - x_{ij}) = 0 \\ ((i,j) \in F^c \times L^c) \\ \sum_{k \in L^c} x_{ik} - x_0 = 0\ (i \in F^c), \\ \sum_{k \in F^c} x_{kj} - x_0 = 0\ (j \in L^c) \end{array} \right\}$$

for some matrix $\boldsymbol{Q}(F,L) \in \mathbb{S}^{\{0,F^c \times L^c\}}$, which is given by $\boldsymbol{Q}(F,L) = \boldsymbol{P}(F,L)^T \boldsymbol{Q}^0 \boldsymbol{P}(F,L)$, where

$$\begin{aligned} \boldsymbol{P}(F,L) &= \text{the } (1 + |N \times N|) \times (1 + |F^c \times L^c|) \text{ matrix with elements} \\ P(F,L)_{\boldsymbol{\alpha}\boldsymbol{\beta}} &= \begin{cases} 1 & \text{if } \boldsymbol{\alpha} = 0 \text{ and } \boldsymbol{\beta} = 0, \\ 1 & \text{if } \boldsymbol{\alpha} = (i_p, j_p) \text{ for some } p \in \{1, \ldots, r\} \text{ and } \boldsymbol{\beta} = 0, \\ 1 & \text{if } \boldsymbol{\alpha} = \boldsymbol{\beta} \in F^c \times L^c, \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

Obviously, $\boldsymbol{P}(\emptyset,\emptyset)$ coincides with the identity matrix in $\mathbb{S}^{\{1,N\times N\}}$ and $\mathrm{QAP}(\emptyset,\emptyset)$ corresponds to QAP (2).

2.3 Enumeration tree

A fundamental step in the B&B method is to successively construct an enumeration tree. Each node of the enumeration tree corresponds to a subproblem of QAP (2), $\mathrm{QAP}(F,L)$ for some $(F,L) \in \Pi_r \times \Pi_r$ with $r \in \{0,1,\ldots,n\}$. Thus, each node is identified with a subproblem. To initialize the tree, we set $\mathrm{QAP}(\emptyset,\emptyset)$ (*i.e.,* QAP (2)) for the root node and compute the incumbent objective value $\hat{\zeta}$ by applying a heuristic method to QAP (2).

Suppose that we have generated a node $\mathrm{QAP}(F,L)$, where $F = (i_1, \ldots, i_r) \in \Pi_r$ and $L = (j_1, \ldots, j_r) \in \Pi_r$ for some $r \in \{1, \ldots, n\}$ (or $(F,L) = (\emptyset,\emptyset)$). Then, we compute a lower bound $\zeta^\ell(F,L)$ (see Section 3) and an upper bound $\zeta^u(F,L)$ (see Section 4), which is associated with a feasible solution of $\mathrm{QAP}(F,L)$, for the (unknown) optimal value $\zeta(F,L)$; $\zeta^\ell(F,L) \le \zeta(F,L) \le \zeta^u(F,L)$. The incumbent objective value $\hat{\zeta}$ is then updated by $\hat{\zeta} = \min\{\hat{\zeta},\ \zeta^u(F,L)\}$.

When $\hat{\zeta} \le \zeta^\ell(F,L)$ occurs, the (unknown) optimal value $\zeta(F,L)$ of $\mathrm{QAP}(F,L)$ is greater than or equal to the incumbent objective value $\hat{\zeta}$. In this case, the node is pruned (or terminated). Otherwise, branching the node into $n-r$ child nodes takes place as follows. We select either facility $f_+ \in F^c$ or location $\ell_+ \in L^c$. If $f_+ \in F^c$ is selected, then $|L^c| = n-r$ child nodes $\mathrm{QAP}((F,f_+),(L,\ell))$ $(\ell \in L^c)$ are generated. Otherwise, $|F^c| = n-r$ child nodes $\mathrm{QAP}((F,f),(L,\ell+))$ $(f \in F^c)$ are generated. A branching rule determines which $f_+ \in F^c$ or $\ell_+ \in L^c$ to select. In Section 5, we describe three types of branching rules in detail.

For any $\mathrm{QAP}(F,L)$ possibly located at the bottom level (or the nth level), we have $|F| = |L| = n$ and $F^c = L^c = \emptyset$. Thus, $\mathrm{QAP}(F,L)$ becomes a trivial QAP with every facility assigned to some location and vice versa. It is trivial to compute its optimal value $\zeta(F,L)$. We note that the height of the enumeration tree is at most n. Numerically, we can enumerate all $(n-r)!$ feasible solutions of $\mathrm{QAP}(F,L)$ to compute its exact optimal value when $|F^c| = |L^c| = n - r \le m$ holds for a small enough m. In the numerical results reported in Section 6, we took $m = 7$.

3 Lower bounding procedure

Throughout this section, we fix $(F,L) \in \Pi_r \times \Pi_r$ $(r \in \{0,1,\ldots,n\})$, and present a method to compute a lower bound for the optimal objective value $\zeta(F,L)$ of $\mathrm{QAP}(F,L)$. Before deriving a doubly nonnegative (DNN) relaxation of $\mathrm{QAP}(F,L)$ in Section 3.1 and a Lagrangian DNN relaxation of $\mathrm{QAP}(F,L)$ in Section 3.2, we first transform $\mathrm{QAP}(F,L)$ into $\overline{\mathrm{QAP}}(F,L)$ described below.

We note that each homogeneous linear equality constraint $\sum_{k \in L^c} x_{ik} - x_0 = 0$ of $\mathrm{QAP}(F,L)$ can be written as $\hat{\boldsymbol{c}}(i,L)^T \boldsymbol{x} = 0$ for some $\hat{\boldsymbol{c}}(i,L) \in \mathbb{R}^{\{0,F^c \times L^c\}}$ $(i \in F^c)$. Let $\hat{\boldsymbol{a}}(i,L) = \mathbf{vec}(\hat{\boldsymbol{c}}(i,L)\hat{\boldsymbol{c}}(i,L)^T)^T$. Then, the

linear equality is equivalent to $\hat{\boldsymbol{a}}(i,L)\mathbf{vec}(\boldsymbol{x}\boldsymbol{x}^T)=0$. Similarly, we can rewrite each homogeneous linear constraint $\sum_{k\in F^c} x_{kj}-x_0=0$ as $\tilde{\boldsymbol{a}}(j,F)\mathbf{vec}(\boldsymbol{x}\boldsymbol{x}^T)=0$ for some $(1+|F^c||L^c|)^2$-dimensional row vector $\tilde{\boldsymbol{a}}(j,F)$ $(j\in L^c)$. Hence, the entire linear constraints

$$\sum_{k\in L^c} x_{ik}-x_0=0\ (i\in F^c)\ \text{ and }\ \sum_{k\in F^c} x_{kj}-x_0=0\ (j\in L^c)$$

can be expressed as $\boldsymbol{A}(F,L)\mathbf{vec}(\boldsymbol{x}\boldsymbol{x}^T)=\mathbf{0}\in\mathbb{R}^{|F^c|+|L^c|}$, where $\boldsymbol{A}(F,L)$ denotes the $(|F^c|+|L^c|)\times(1+|F^c||L^c|)^2$ matrix consisting of the row vectors $\hat{\boldsymbol{a}}(i,L)$ $(i\in F^c)$ and $\tilde{\boldsymbol{a}}(j,F)$ $(j\in L^c)$.

Define

$$\begin{aligned}
\mathbb{K}_1(F,L) &= \mathbb{S}_+^{\{0,F^c\times L^c\}},\\
\mathbb{K}_2(F,L) &= \left\{\boldsymbol{X}\in\mathbb{S}^{\{0,F^c\times L^c\}}:\ \begin{array}{l} X_{\boldsymbol{\alpha}\boldsymbol{\beta}}\ge 0\ (\boldsymbol{\alpha},\boldsymbol{\beta}\in\{0,F^c\times L^c\}),\\ X_{0\boldsymbol{\alpha}}-X_{\boldsymbol{\alpha}\boldsymbol{\alpha}}=0\ (\boldsymbol{\alpha}\in F^c\times L^c)\end{array}\right\},\\
\boldsymbol{H}(F,L) &= \text{the matrix in } \mathbb{S}^{\{0,F^c\times L^c\}} \text{ with 1 at the } (0,0)\text{th element}\\
&\quad \text{and 0 elsewhere.}
\end{aligned}$$

By construction, the constraint $x_{ij}(x_0-x_{ij})=0$ $((i,j)\in F^c\times L^c)$ is equivalent to $X_{0\boldsymbol{\alpha}}-X_{\boldsymbol{\alpha}\boldsymbol{\alpha}}=0$ $(\boldsymbol{\alpha}\in F^c\times L^c)$ if $\boldsymbol{X}=\boldsymbol{x}\boldsymbol{x}^T\in\mathbb{S}^{\{0,F^c\times L^c\}}$ and $\boldsymbol{x}\in\mathbb{R}^{\{0,F^c\times L^c\}}$, and $\langle\boldsymbol{H}(F,L),\,\boldsymbol{X}\rangle=X_{00}$ for every $\boldsymbol{X}\in\mathbb{S}^{\{0,F\times L\}}$. Therefore, we can rewrite QAP(F,L) given in Section 2.2 as

$$\overline{\mathrm{QAP}}(F,L):\zeta(F,L)=\left\{\langle\boldsymbol{Q}(F,L),\,\boldsymbol{X}\rangle:\ \begin{array}{l}\boldsymbol{x}\in\mathbb{R}^{\{0,F^c\times L^c\}},\ \boldsymbol{X}=\boldsymbol{x}\boldsymbol{x}^T\in\mathbb{K}_1(F,L)\cap\mathbb{K}_2(F,L),\\ \langle\boldsymbol{H}(F,L),\,\boldsymbol{X}\rangle=1,\ \boldsymbol{A}(F,L)\mathbf{vec}(\boldsymbol{X})=\mathbf{0}\end{array}\right\}.$$

3.1 A DNN relaxation of $\overline{\mathrm{QAP}}(F,L)$

In $\overline{\mathrm{QAP}}(F,L)$, the constraint $\boldsymbol{X}=\boldsymbol{x}\boldsymbol{x}^T$ requires the rank of the variable matrix $\boldsymbol{X}$ to be 1. By removing this constraint, we obtain

$$\mathrm{DNN}^p(F,L):\eta^p(F,L) \;=\; \min\left\{\langle\boldsymbol{Q}(F,L),\,\boldsymbol{X}\rangle:\ \begin{array}{l}\boldsymbol{X}\in\mathbb{K}_1(F,L)\cap\mathbb{K}_2(F,L),\\ \langle\boldsymbol{H}(F,L),\,\boldsymbol{X}\rangle=1,\ \boldsymbol{A}(F,L)\mathbf{vec}(\boldsymbol{X})=\mathbf{0}\end{array}\right\}.$$

which serves as a DNN relaxation of $\overline{\mathrm{QAP}}(F,L)$ with $\eta^p(F,L)\le\zeta(F,L)$.

3.2 A Lagrangian DNN relaxation problem of $\overline{\mathrm{QAP}}(F,L)$

We know that

$$\boldsymbol{A}(F,L)\mathbf{vec}(\boldsymbol{X})\ge\mathbf{0}\text{ for every }\boldsymbol{X}\in\mathbb{K}_1(F,L)=\mathbb{S}_+^{\{0,F^c\times L^c\}},$$

since each row vector of $\boldsymbol{A}(F,L)$ is of the form $\mathbf{vec}(\boldsymbol{c}\boldsymbol{c}^T)$ for some $\boldsymbol{c}\in\mathbb{R}^{\{0,F^c\times L^c\}}$. Hence, the constraint $\boldsymbol{A}(F,L)\mathbf{vec}(\boldsymbol{X})=\mathbf{0}$ in $\mathrm{DNN}^p(F,L)$ can be replaced by

$$0 \;=\; \boldsymbol{e}^T\boldsymbol{A}(F,L)\mathbf{vec}(\boldsymbol{X}) \;=\; \Big\langle\sum_{i\in F^c}\hat{\boldsymbol{c}}(i,L)\hat{\boldsymbol{c}}(i,L)^T+\sum_{j\in L^c}\tilde{\boldsymbol{c}}(j,F)\tilde{\boldsymbol{c}}(i,L)^T,\,\boldsymbol{X}\Big\rangle,$$

where $\boldsymbol{e}\in\mathbb{R}^{|F|+|L|}$ denotes the vector of 1's, and the second equality holds by the construction of $\boldsymbol{A}(F,L)$. Applying the Lagrangian relaxation to the resulting problem, we obtain

$$\mathrm{DNN}^p_\lambda(F,L):\eta^p_\lambda(F,L) \;=\; \min\left\{\langle\boldsymbol{Q}_\lambda(F,L),\,\boldsymbol{X}\rangle:\ \begin{array}{l}\boldsymbol{X}\in\mathbb{K}_1(F,L)\cap\mathbb{K}_2(F,L),\\ \langle\boldsymbol{H}(F,L),\,\boldsymbol{X}\rangle=1\end{array}\right\},$$

and its dual

$$\mathrm{DNN}^d_\lambda(F,L):\eta^d_\lambda(F,L) \;=\; \max\left\{y\in\mathbb{R}:\ \begin{array}{l}\boldsymbol{Y}_1\in\mathbb{K}_1(F,L)^*,\ \boldsymbol{Y}_2\in\mathbb{K}_2(F,L)^*,\\ \boldsymbol{Q}_\lambda(F,L)-\boldsymbol{H}(F,L)y=\boldsymbol{Y}_1+\boldsymbol{Y}_2\end{array}\right\}.$$

Here $\lambda \in \mathbb{R}$ denotes a Lagrangian multiplier, and

$$\boldsymbol{Q}_\lambda(F,L) \quad = \quad \boldsymbol{Q}(F,L) + \lambda \left(\sum_{i \in F^c} \hat{\boldsymbol{c}}(i,L)\hat{\boldsymbol{c}}(i,L)^T + \sum_{j \in L^c} \tilde{\boldsymbol{c}}(j,F)\tilde{\boldsymbol{c}}(i,L)^T \right) \in \mathbb{S}^{\{0,F^c \times L^c\}}.$$

$\mathrm{DNN}^p_\lambda(F,L)$ serves as a Lagrangian relaxation of $\mathrm{DNN}^p(F,L)$ and a Lagrangian DNN relaxation of $\overline{\mathrm{QAP}}(F,L)$ with $\eta^p_\lambda(F,L) \leq \eta^p(F,L) \leq \zeta(F,L)$ for any $\lambda \in \mathbb{R}$. We can prove that $\eta^p_\lambda(F,L)$ converges monotonically to $\eta^p(F,L)$ as $\lambda \to \infty$. We also see by the duality theorem ([2, Lemma 2.3]) that $\eta^p_\lambda(F,L) = \eta^d_\lambda(F,L)$.

3.3 The Newton-bracketing method for solving the primal-dual pair of $\mathrm{DNN}^p_\lambda(F,L)$ and $\mathrm{DNN}^d_\lambda(F,L)$

To describe the NB method for solving $\mathrm{DNN}^p_\lambda(F,L)$ and $\mathrm{DNN}^d_\lambda(F,L)$, λ is fixed to a sufficiently large positive number, say $\lambda = 100,000$. The optimal value η^d_λ of $\mathrm{DNN}^d_\lambda(F,L)$ is denoted by y^* throughout this subsection.

Let $-\infty = \mathrm{lb}^0 < y^* < \mathrm{ub}^0$. The NB method applied to $\mathrm{DNN}^p_\lambda(F,L)$ and $\mathrm{DNN}^d_\lambda(F,L)$ generates

$$\begin{aligned}
&\widehat{\boldsymbol{X}}^k \in \mathbb{K}_1(F,L) \cap \mathbb{K}_2(F,L),\ (y^k, \widehat{\boldsymbol{Y}}^k_1, \widehat{\boldsymbol{Y}}^k_2) \in \mathbb{R} \times \mathbb{S}^{\{0,F^c \times L^c\}} \times \mathbb{K}_2(F,L)^*, \\
&\mathrm{lb}^k \in \mathbb{R},\ \mathrm{ub}^k \in \mathbb{R},\ 0 \geq \tau^k \in \mathbb{R}
\end{aligned}$$

$(k = 1, 2, \ldots)$ such that

$$\left.\begin{array}{l}
\boldsymbol{Q}(F,L) - \boldsymbol{H}(F,L)y^k - \widehat{\boldsymbol{Y}}^k_1 - \widehat{\boldsymbol{Y}}^k_2 = \boldsymbol{O}, \\
\mathrm{lb}^k = y^k + (1 + |F^c|)\tau^k,\ \tau^k = \min\{0, \text{the minimum eigenvalue of } \widehat{\boldsymbol{Y}}^k_1\}, \\
\mathrm{lb}^0 \leq \mathrm{lb}^k \leq \mathrm{lb}^{k+1} \to y^* \leftarrow \mathrm{ub}^{k+1} \leq \mathrm{ub}^k \leq \mathrm{ub}^0, \\
\boldsymbol{X} = \widehat{\boldsymbol{X}}^k \text{ is a feasible solution of } \mathrm{DNN}^p_\lambda(F,L),\ \langle \boldsymbol{Q}(F,L), \widehat{\boldsymbol{X}}^k \rangle = \mathrm{ub}^k.
\end{array}\right\} \tag{3}$$

We see from these relations that for sufficiently large k, $\boldsymbol{X} = \widehat{\boldsymbol{X}}^k$ and $(y, \boldsymbol{Y}_1, \boldsymbol{Y}_2) = (y^k, \widehat{\boldsymbol{Y}}^k_1, \widehat{\boldsymbol{Y}}^k_2)$ provides approximate optimal solutions of $\mathrm{DNN}^p_\lambda(F,L)$ and $\mathrm{DNN}^d_\lambda(F,L)$, respectively. Hence $\boldsymbol{X} = \widehat{\boldsymbol{X}}^k$ is an approximate optimal solution of $\mathrm{DNN}^p(F,L)$. See [10] for more details.

It is natural to use a stopping criterion for the NB method as

(a) $|\mathrm{ub}^k - \mathrm{lb}^k| < \epsilon \max\{|\mathrm{lb}^k|, |\mathrm{ub}^k|, 1\}$, where ϵ denotes a sufficiently small positive number.

When the NB method is implemented in the B&B method for QAPs with integer optimal values, an accurate approximation of y^* does not have to be computed all the time by the NB method. We can terminate the iteration in case when

(b) $\hat{\zeta} \leq \lceil \mathrm{lb}^k \rceil$ or

(c) $\mathrm{ub}^k < \hat{\zeta}$,

where $\hat{\zeta}$ denotes an integer incumbent objective value of the root node QAP. If (b) occurs, then we know that the sub QAP, $\overline{\mathrm{QAP}}(F,L)$ to which the NB method has applied, does not have any feasible solution whose objective value is smaller than the incumbent objective value $\hat{\zeta}$; hence $\overline{\mathrm{QAP}}(F,L)$ can be pruned. If case (c) occurs, then $y^* \leq \mathrm{ub}^k < \hat{\zeta}$. Thus, branching $\overline{\mathrm{QAP}}(F,L)$ further is necessary even if the NB iteration continues. By employing these two additional criteria (b) and (c), we can save a lot of computational time for the NB method. This is a distinctive advantage of using the NB method in the B&B method.

4 Upper bounding procedures

Let $(F,L) \in \Pi_r \times \Pi_r$ for some $r \in \{0, 1, \ldots, n\}$. An approximate optimal solution of $\mathrm{QAP}(F,L)$ needs to be computed for the upper bounding procedure. A simple rounding of an approximate optimal solution of its Lagrangian relaxation $\mathrm{DNN}^p_\lambda(F,L)$ and the tabu search method [22, 23] for $\mathrm{QAP}(F,L)$ have been used for the computation. We plan to replace this method by the one presented below in this section to effectively utilize information from $\mathrm{DNN}^p_\lambda(F,L)$. In the remainder of this section, we assume $F = L = \emptyset$

so that QAP(F, L) coincides with the root node QAP (2), for simplicity of notation. All the discussions, however, can be easily adapted to any sub QAP, QAP(F, L).

Suppose that we have applied the NB method to the primal-dual pair of DNN problems DNN$^p_\lambda(\emptyset, \emptyset)$ and DNN$^d_\lambda(\emptyset, \emptyset)$ and obtained an approximate optimal solution $\widehat{\boldsymbol{X}}$ of DNN$^p_\lambda(\emptyset, \emptyset)$. The set of feasible solutions $\boldsymbol{X}$ of DNN$^p_\lambda(\emptyset, \emptyset)$ may be regarded as a doubly nonnegative relaxation of the set of $\boldsymbol{u}\boldsymbol{u}^T$ with feasible solutions $\boldsymbol{u}$ of QAP (2), and the first column of $\widehat{\boldsymbol{X}}$ corresponds to a feasible (or approximate optimal) solution $u_0\boldsymbol{u} = \boldsymbol{u}$ of QAP (2). We know that $\boldsymbol{U} = [u_{ik}]$ forms a permutation matrix for every feasible solution $\boldsymbol{u}$ of QAP (2). Through preliminary numerical experiment, we have observed that the $n \times n$ matrix

$$\widehat{\boldsymbol{U}} \;=\; \begin{pmatrix} \widehat{X}_{11,0} & \widehat{X}_{12,0} & \cdots & \widehat{X}_{1n,0} \\ \widehat{X}_{21,0} & \widehat{X}_{22,0} & \cdots & \widehat{X}_{2n,0} \\ \cdots & \cdots & \cdots & \cdots \\ \widehat{X}_{n1,0} & \widehat{X}_{n2,0} & \cdots & \widehat{X}_{nn,0} \end{pmatrix}$$

approximately forms a doubly-stochastic matrix. A popular approach for recovering a feasible solution from $\widehat{\boldsymbol{U}}$ is through appropriate rounding procedures. Here we propose a method to compute a permutation matrix $\boldsymbol{U} = \overline{\boldsymbol{U}}$ which minimize the distance $\| \boldsymbol{U} - \widehat{\boldsymbol{U}} \|$ among all permutation matrices $\boldsymbol{U}$:

$$\begin{aligned} \bar{\zeta} &= \min \left\{ \| \boldsymbol{U} - \widehat{\boldsymbol{U}} \|^2 \colon \boldsymbol{U} \in \mathbb{R}^{n\times n} \text{ is a permutation matrix} \right\} \\ &= \min \left\{ -2\langle \widehat{\boldsymbol{U}}, \boldsymbol{U} \rangle + \| \widehat{\boldsymbol{U}} \|^2 + n^2 : \boldsymbol{U} \in \mathbb{R}^{n\times n} \text{ is a permutation matrix} \right\} \\ &\quad (\text{since } \| \boldsymbol{U} \|^2 = n^2 \text{ for every permutation matrix } \boldsymbol{U} \in \mathbb{R}^{n\times n}). \\ &= \min \left\{ \langle -2\widehat{\boldsymbol{U}}, \boldsymbol{U} \rangle : \boldsymbol{U} \in \mathbb{R}^{n\times n} \text{ is a permutation matrix} \right\} + \| \widehat{\boldsymbol{U}} \|^2 + n^2, \end{aligned}$$

where $\| \cdot \|$ denotes the Frobenius norm. This problem forms a linear assignment problem, which can be solved the well-known Hungarian method [12].

5 Branching rules

Let $(F, L) \in \Pi_r \times \Pi_r$ for some $r \in \{0, 1, \ldots, n\}$. As we have briefly mentioned in Section 2.3, if node QAP(F, L) is not pruned, we select either an $f_+ \in F^c$ to branch QAP(F, L) into $|L^c|$ child nodes QAP$((F, f_+), (L, \ell))$ $(\ell \in L^c)$ or an $\ell_+ \in L^c$ to branch QAP(F, L) into $|F^c|$ child nodes QAP$((F, f), (L, \ell_+))$ $(f \in F^c)$.

We present three types of rules for selecting an $f_+ \in F^c$ or an $\ell_+ \in L^c$. In each rule, an evaluation function $\varphi(f, \ell)$ is introduced for possible child nodes QAP$((F, f), (L, \ell))$ $((f, \ell) \in F^c \times L^c)$ so that $\varphi(f, \ell)$ can represent the optimal values of QAP$((F, f), (L, \ell))$ or its Lagrangian DNN relaxation problem DNN$^p_\lambda((F, f), (L, \ell))$. Then, the following functions are defined:

$$\begin{aligned} \bar{\varphi}(f, \cdot) &= \frac{1}{|L^c|} \sum_{\ell \in L^c} \varphi(f, \ell) \;\; (\text{the mean of } \varphi(f, \ell) \text{ over } \ell \in L^c) \; (f \in F^c), \\ \bar{\varphi}(\cdot, \ell) &= \frac{1}{|F^c|} \sum_{f \in F^c} \varphi(f, \ell) \;\; (\text{the mean of } \varphi(f, \ell) \text{ over } f \in F^c) \; (\ell \in L^c). \end{aligned}$$

Our method is to choose $f_+ = \arg\max\{\bar{\varphi}(f, \cdot) : f \in F^c\}$ if $\max\{\bar{\varphi}(f, \cdot) : f \in F^c\} \geq \max\{\bar{\varphi}(\cdot, \ell) : \ell \in L^c\}$, and $\ell_+ = \arg\max\{\bar{\varphi}(\cdot, \ell) : \ell \in L^c\}$ otherwise. Each rule employs a different function $\varphi(f, \ell)$ from others. We describe how $\varphi(f, \ell)$ is defined with a fixed $((f, \ell) \in F^c \times L^c)$ for the three rules in Sections 5.1, 5.2 and 5.2, respectively.

5.1 Branching rule M: a rule using the mean of all feasible objective values of $\mathbf{Q}(F, L)$ $((F, L) \in \Pi_r \times \Pi_r,\ r = 0, 1, \ldots, n)$

Define $\varphi(f, \ell)$ to be the mean of the objective values of QAP$((F, f), (L, \ell))$ over its feasible solutions, which can be computed by substituting $x_0 = 1$ and $x_{ij} = 1/(n - r - 1)$ $((i, j) \in (F, f)^c \times (L, \ell)^c)$ into the objective function $\langle \boldsymbol{Q}((F, f), (L, \ell)), \boldsymbol{x}\boldsymbol{x}^T \rangle$ of QAP$((F, f), (L, \ell))$. This branching rule M is employed in the current parallel implementation of the B&B method on the UG framework.

5.2 Branching rule P: a rule using an approximate optimimal solution $\widehat{\boldsymbol{X}}$ of $\mathbf{DNN}^p_\lambda(F, L)$

Let $\widehat{\boldsymbol{X}}$ be an approximate optimal solution of $\mathrm{DNN}^p_\lambda(F, L)$. Then, $\widehat{\boldsymbol{X}}$ may be regarded as an approximate optimal solution of $\mathrm{DNN}^p(F, L)$ since $\mathrm{DNN}^p_\lambda(F, L)$ is a Lagrangian relaxation of $\mathrm{DNN}^p(F, L)$ and its optimal value $\eta^p_\lambda(F, L)$ converges to $\eta^p(F, L)$ of $\mathrm{DNN}^p(F, L)$ as $\lambda \to \infty$. We will construct a rough approximate optimal solution $\widehat{\boldsymbol{X}}(f, \ell)$ of $\mathrm{DNN}^p((F, f), (L, \ell))$ from $\widehat{\boldsymbol{X}}$, and define $\varphi(f, \ell)$ to be the objective value of $\mathrm{DNN}^p((F, f), (L, \ell))$ at $\widehat{\boldsymbol{X}}(f, \ell)$, *i.e.*, $\varphi(f, \ell) = \langle \boldsymbol{Q}((F, f), (L, \ell)), \widehat{\boldsymbol{X}}(f, \ell)\rangle$ as described below.

We first project $\widehat{\boldsymbol{X}} \in \mathbb{S}^{\{0,F^c\times L^c\}}$ onto the linear space $\mathbb{S}^{\{0,(F,f)^c\times(L,\ell)^c\}}$ where the feasible region of $\mathrm{QAP}^p((F, f), (L, \ell))$ lies. Let $\boldsymbol{X}(f, \ell)$ denote the metric projection of $\widehat{\boldsymbol{X}}$, which is obtained by deleting the rows $\widehat{\boldsymbol{X}}_{i.}$ $(i \notin \{0, (F, f)^c \times (L, \ell)^c\})$ and the columns $\widehat{\boldsymbol{X}}_{.j}$ $(j \notin \{0, (F, f)^c \times (L, \ell)^c\})$ from $\widehat{\boldsymbol{X}}$ simultaneously. We also know that every feasible solution $\boldsymbol{X}$ of $\mathrm{DNN}^p((F, f), (L, \ell))$ must lie in the linear manifold

$$M \;=\; \left\{ \boldsymbol{X} \in \mathbb{S}^{\{0,(F,f)^c\times(L,\ell)^c\}} : \begin{array}{l} X_{00} = 1,\ \boldsymbol{A}((F, f), (L, \ell))\mathbf{vec}(\boldsymbol{X}) = \boldsymbol{0}, \\ X_{0\boldsymbol{\alpha}} - X_{\boldsymbol{\alpha}\boldsymbol{\alpha}} = 0\ (\boldsymbol{\alpha} \in (F, f)^c \times (L, \ell)^c) \end{array} \right\}.$$

Let $\widehat{\boldsymbol{X}}(f, \ell)$ be the metric projection of $\boldsymbol{X}(f, \ell)$ onto M, and then define

$$\varphi(f, \ell) = \langle \boldsymbol{Q}((F, f), (L, \ell)), \widehat{\boldsymbol{X}}(f, \ell)\rangle.$$

We note that $\widehat{\boldsymbol{X}}(f, \ell)$ is not necessarily in $\mathbb{K}_1((F, f), (L, \ell)) = \mathbb{S}^{\{0,(F,f)^c\times(L,\ell)^c\}}_+$; hence $\widehat{\boldsymbol{X}}(f, \ell)$ is not necessarily a feasible solution of $\mathrm{DNN}^p((F, f), (L, \ell))$ in general.

5.3 Branching rule D: a rule using an approximate optimal solution $(\hat{y}, \widehat{\boldsymbol{Y}}_1, \widehat{\boldsymbol{Y}}_2)$ of $\mathbf{DNN}^d_\lambda(F, L)$

Recall that the NB method applied to the primal-dual pair of $\mathrm{DNN}^p_\lambda(F, L)$ and $\mathrm{DNN}^d_\lambda(F, L)$ generates

$$\begin{array}{l} \boldsymbol{X}^k \in \mathbb{K}_1(F, L) \cap \mathbb{K}_2(F, L),\ (y^k, \boldsymbol{Y}^k_1, \boldsymbol{Y}^k_2) \in \mathbb{R} \times \mathbb{S}^{\{0,F\times L\}} \times \mathbb{K}_2(F, L)^*, \\ \mathrm{lb}^k \in \mathbb{R},\ \mathrm{ub}^k \in \mathbb{R},\ 0 \geq \tau^k \in \mathbb{R} \end{array}$$

$(k = 1, 2, \ldots)$ satisfying (3). We assume that the method terminates with the stopping criterion (c) $\mathrm{ub}^k < \hat{\zeta}$, which requires to branch the node $\mathrm{QAP}(F, L)$.

Define the $(1 + |F^c \times L^c|) \times (1 + |(F, f)^c \times (L, \ell)^c|)$ matrix $\boldsymbol{P}(f, \ell)$ by

$$P(f, \ell)_{\boldsymbol{\alpha}\boldsymbol{\beta}} \;=\; \left\{ \begin{array}{ll} 1 & \text{if } \boldsymbol{\alpha} = 0 \text{ and } \boldsymbol{\beta} = 0, \\ 1 & \text{if } \boldsymbol{\alpha} = (f, \ell) \text{ and } \boldsymbol{\beta} = 0, \\ 1 & \text{if } \boldsymbol{\alpha} = \boldsymbol{\beta} \in (F, f)^c \times (L, \ell)^c, \\ 0 & \text{otherwise .} \end{array} \right.$$

Then

$$\begin{array}{l} \boldsymbol{Q}((F, f), (L, \ell)) = \boldsymbol{P}(f, \ell)^T \boldsymbol{Q}(F, L)\boldsymbol{P}(f, \ell) \in \mathbb{S}^{\{0,(F,f)\times(L,\ell)\}}, \\ \boldsymbol{H}((F, f), (L, \ell)) = \boldsymbol{P}(f, \ell)^T \boldsymbol{H}(F, L)\boldsymbol{P}(f, \ell) \in \mathbb{S}^{\{0,(F,f)\times(L,\ell)\}}, \\ \widehat{\boldsymbol{Y}}_1(f, \ell) \equiv \boldsymbol{P}(f, \ell)^T \boldsymbol{Y}^k_1 \boldsymbol{P}(f, \ell) \in \mathbb{S}^{\{0,(F,f)^c\times(L,\ell)^c\}}, \\ \widehat{\boldsymbol{Y}}_2(f, \ell) \equiv \boldsymbol{P}(f, \ell)^T \boldsymbol{Y}^k_2 \boldsymbol{P}(f, \ell) \in \mathbb{K}_2(F, f), (L, \ell))^* \end{array}$$

hold. We apply the above to the identity $\boldsymbol{Q}(F, L) - \boldsymbol{H}(F, L)y^k - \boldsymbol{Y}^k_1 - \boldsymbol{Y}^k_2 = \boldsymbol{O}$ in (3). More precisely, multiplying $\boldsymbol{P}((F, f), (L, \ell))^T$ to each term of the identity on the left and $\boldsymbol{P}((F, f), (L, \ell))$ on the right results in

$$\begin{array}{l} \boldsymbol{O} \;=\; \boldsymbol{Q}((F, f), (L, \ell)) - \boldsymbol{H}((F, f), (L, \ell))y^k - \widehat{\boldsymbol{Y}}_1(f, \ell) - \widehat{\boldsymbol{Y}}_2(f, \ell), \\ \widehat{\boldsymbol{Y}}_1(f, \ell) \in \mathbb{S}^{\{0,(F,f)^c\times(L,\ell)^c\}},\ \widehat{\boldsymbol{Y}}_2(f, \ell) \in \mathbb{K}_2(F, f), (L, \ell))^*. \end{array}$$

Now, define

$$\begin{aligned}\tau(f,\ell) &= \min\{0, \text{the minimum eigenvalue of } \widehat{\boldsymbol{Y}}_1(f,\ell)\},\\ \varphi(f,\ell) &= y^k + (1+|(F,f)^c|)\tau(f,\ell) \;=\; y^k + (n-r)\tau(f,\ell).\end{aligned}$$

Then we can prove that $\mathrm{lb}^k \le \varphi(f,\ell) \le \eta^p_\lambda((F,f),(L,\ell))$, which implies that $\varphi(f,\ell)$ itself serves as a lower bound of the optimal value $\zeta((F,f),(L,\ell))$ of the child node $\mathrm{QAP}((F,f),(L,\ell))$. Thus, if $\hat{\zeta} \le \varphi(f,\ell)$ holds, then we can terminate $\mathrm{QAP}((F,f),(L,\ell))$ without applying the NB method to $\mathrm{QAP}((F,f),(L,\ell))$. This is a distinct advantage of the branching rule **D**, although the bound $\varphi(f,\ell)$ may not be as strong as $\eta^p_\lambda((F,f),(L,\ell))$.

6 Preliminary numerical results

We have implemented two types of codes for the parallel B&B method to solve QAPs. The first one is written in MATLAB, which has been developed to see how effectively and efficiently the NB method works in the branch-and-bound framework. The code used there for the NB method is modifications of the ones written for the MATLAB software package NewtBracket [11], which was released very recently. Specifically, the stopping criteria of the NB method were changed as presented in the last paragraph of Section 3.3. All techniques presented in Sections 2, 3 and 5 have been incorporated in the MATLAB code, but not the upper bounding procedure described in Section 4. The MATLAB code works in parallel but can solve only small-sized QAP instances as shown in Section 6.1.

The second code is a C++ implementation of the MATLAB code. Based on the information and knowledge obtained from preliminary numerical experiment by the MATLAB code, we have been developing the C++ code running on the UG, a generic framework to parallelize branch-and-bound based solvers. An application of the UG can be developed on PC, in which communication is carried out through shared memory [21]. After recompiling the application code with additional small amount of MPI related code for transferring objects, the parallel solver could potentially run with more than 100,000 cores [18, 24].

Two types of processes exist when running the parallelized QAP solver by UG on distributed memory environment. First, there is a single LoadCoordinator, which makes all decisions concerning the dynamic load balancing and distributes subproblems of QAP instances to be solved. Second, all other processes are Solvers that solve the distributed subproblem by regarding it as root node. The UG tries to solve all open subproblems on the single branch-and-bound search tree in parallel as much as it can by using all available Solvers. More precisely, the UG executes dynamic load balancing so that subproblems of the branch-and-bound search tree are transferred to the other Solvers when an idle Solver exists. In order to reduce the idle time of the Solvers, LoadCoordinator tries to keep a small number of open subproblems (the initial number can be specified by a runtime parameter and it is changed dynamically during the computation) so that it can assign a subproblem at the earliest time when an idle Solver exists.

For *ramp-up*, a phase until all cores become busy (see [15]), several methods exist. In our case of the parallelized QAP solver, we used the *normal ramp-up*, which performs the normal parallel branch-and-bound procedure as mentioned above during the ramp-up phase [21]; See also Section 7.1. Until now, we have succeeded to solve tai30a, tai35b, tai40b and sko42 from QAPLIB [7] as reported in Section 6.2. But the C++ code needs further improvements for larger scale QAP instances.

6.1 Small scale instances

The main features of the parallel B&B method implemented in the MATLAB code can be summarized as follows:

- The enumeration tree described in Section 2.3.
- The Lagrangian DNN relaxation of sub QAPs and a modified version of NewtBracket [11]. See Section 3.
- A simple rounding of approximate optimal solutions of $\mathrm{DNN}^p_\lambda(F,L)$ ($(F,L) \in \Pi_r \times \Pi_r$, $r \in \{0,1,\ldots,n\}$) for the upper bounding procedure.
- The three types of branching rules, M, P and D, which are described in Sections 5.1, 5.2 and 5.3, respectively.

- A parallel breadth first search with processing each node QAP(F, L) to compute $\zeta^l(F, L)$ and $\zeta^u(F, L)$ by one core.

The optimal values of all small QAP instances solved are known. The initial incumbent objective value is set to be the optimal value + 1, and it is updated to the optimal value which is found by the upper bound procedure at some iteration. With this setting, a high quality incumbent objective value close to the optimal value is known at the beginning of the B&B method. As a result, the role of the upper bounding procedure becomes less important, and the breadth first search is reasonable for parallel computation.

All the computations for numerical results reported in Table 1, 2 and 3 were performed using MATLAB 2020b on iMac Pro with Intel Xeon W CPU (3.2 GHZ), 8 cores and 128 GB memory.

Numerical results on nug20 $\sim$ nug28, tai20a $\sim$ tai30b and bur20a $\sim$ bur20c are shown in Tables 1, 2 and 3, respectively. We observe that:

- The number of nodes (=the number of sub QAPs processed) greatly depends on the branching rules M, P and D. Overall, the rule D performs better than the others.
- In particular, for bur26b and bur26c instances, the number of nodes generated with the branching rule D is much smaller than those generated by the others.
- But, for the largest size QAP, tai30b, the branching rule M generates the smallest number of nodes.

We describe in detail on which branching rule is preferable for a given larger QAP instance in Section 7.2.

Table 1: Numerical results on small scale QAP instances — 1. Br : Branching rules presented in Section 5.

QAP instance	Opt.val	Br	No. of nodes generated	Total execution time(sec) in para.	Time(sec) for computing LB	UB	Br	No. of CPU cores used
nug20	2,570	M	757	6.78e2	4.21e3	3.13e1	3.38e0	8
		P	301	3.18e2	1.80e3	1.45e1	1.18e1	
		D	412	3.84e2	2.15e3	1.91e1	3.11e1	
nug21	2,438	M	312	4.93e2	2.67e3	1.80e1	1.48e0	8
		P	200	3.46e2	1.73e3	9.72e0	1.04e1	
		D	272	3.73e2	1.75e3	1.33e1	2.59e1	
nug22	3,596	M	429	7.16e2	4.43e3	2.55e1	3.37e0	8
		P	250	4.33e2	1.88e3	1.27e1	1.80e1	
		D	149	4.10e2	2.08e3	9.96e0	2.36e1	
nug24	3,488	M	1,413	3.34e3	2.24e4	9.30e1	2.16e1	8
		P	742	1.99e3	1.28e4	5.02e1	1.06e2	
		D	913	2.57e3	1.46e4	7.39e1	2.46e2	
nug25	3,744	M	8,446	2.00e4	1.49e5	5.60e2	1.53e2	8
		P	3,004	8.37e3	5.75e4	2.14e2	5.57e2	
		D	3,805	8.97e3	6.46e4	3.41e2	1.29e3	
nug27	5,234	M	2,810	1.48e4	1.06e5	4.33e2	1.27e2	8
		P	979	5.26e3	3.29e4	1.32e2	2.78e2	
		D	2,170	9.83e3	6.91e4	3.90e2	1.41e3	
nug28	5,166	M	10,977	5.58e4	4.31e5	2.27e3	5.79e2	8
		P	4,236	2.25e4	1.68e5	8.40e2	1.57e3	
		D	9,977	4.55e4	3.37e5	2.15e3	7.83e3	

6.2 Challenging large scale instances

We solved challenging large scale instances, nug30, tai30a, tai35b, tai40b and sko42 on the ISM (Institute of Statistical Mathematics) supercomputer HPE SGI 8600, which is a liquid cooled, tray-based, high-density clustered computer system. The ISM supercomputer has 384 computing nodes and each node has two Intel Xeon Gold 6154 3.0GHz CPUs (36 cores) and 384GB of memory. The LoadCoordinator process is assigned to a CPU core and each Solver process is assigned to a CPU core. Therefore, the number of Solvers is less than the number of cores by one. All of the instances were solved as a single job, though check-pointing procedures were performed every 30 minutes (see [19] about check-pointing and

Table 2: Numerical results on small scale QAP instances — 2. Br : Branching rules presented in Section 5.

QAP instance	Opt.val	Br	No. of nodes generated	Total execution time(sec) in para.	Time(sec) for computing			No. of CPU cores used
					LB	UB	Br	
tai20a	703,482	M	2,444	1.49e3	1.04e4	1.06e2	7.72e-1	8
		P	2,107	1.42e3	9.18e3	9.23e1	7.30e1	
		D	1,600	1.17e3	8.30e3	7.31e1	1.02e2	
tai20b	122,455,319	M	183	2.03e2	2.34e2	1.13e0	3.12e-1	8
		P	183	2.01e2	2.32e2	1.19e0	2.22e0	
		D	146	2.40e2	2.38e2	1.23e0	6.11e0	
tai25b	344,355,646	M	367	9.68e2	2.76e3	1.18e1	2.54e-1	8
		P	367	1.08e3	3.20e3	1.31e1	2.41e1	
		D	367	1.08e3	2.90e3	1.26e1	6.21e1	
tai30b	637,117,113	M	989	1.11e4	7.03e4	3.87e2	6.83e1	8
		P	1,654	1.87e4	1.06e5	3.97e2	6.26e2	
		D	1,150	1.80e4	9.74e4	4.01e2	1.51e3	

Table 3: Numerical results on small scale QAP instances — 3. Br : Branching rules presented in Section 5.

QAP instance	Opt.val	Br	No. of nodes generated	Total execution time(sec) in para.	Time(sec) for computing			No. of CPU cores used
					LB	UB	Br	
bur26a	5,426,670	M	2,976	4.21e3	2.03e4	2.46e1	2.13e1	8
		P	11,401	1.70e4	9.24e4	7.69e1	4.00e2	
		D	2,358	3.35e3	1.52e4	1.80e1	1.81e2	
bur26b	3,817,852	M	144,228	4.62e4	2.46e5	1.02e3	1.32e2	8
		P	More than 544,719	More than 3 days				
		D	8,625	5.15e3	2.51e4	7.49e1	2.42e2	
bur26c	5,426,795	M	16,990	6.66e3	3.18e4	1.20e2	2.12e1	8
		P	66,428	8.84e4	5.08e5	3.82e2	1.78e3	
		D	2,416	3.06e3	9.90e3	1.71e1	9.59e1	

restart mechanism). Table 4 shows the computational results. Note that tai30a and sko42 were solved to the optimality for the first time.

Table 4: Numerical results on challenging large scale QAP instances. Branching rule used was M.

QAP instance	Opt.val	No. of nodes generated	Total execution time(sec) in para.	No. of CPU cores used
nug30	6,124	26,181	3.14e3	1,728
tai30a	1,818,146	34,000,579	5.81e5	1,728
tai35b	283,315,445	2,620,547	2.49e5	1,728
tai40b	637,250,948	278,465	1.05e5	1,728
sko42	15,812	6,019,419	5.12e5	5,184

7 Some additional techniques for further developments

7.1 A complete enumeration tree generated by the simple branching rule M

A distinctive feature of the branching rule M presented in Section 5.1 is the independence from any lower bounding and upper bounding procedures. Therefore, we could create a complete enumeration tree using this rule in advance to start the B&B method. We usually start with the single root node associated with the original problem to be solved even if it is implemented in parallel.

At the beginning of the B&B method with the normal ramp-up, only one SOLVER is used, and all others are idle till it finishes processing the original problem at the root node. As the B&B method proceeds, the number of SOLVERs to solve subproblems increases gradually. In the early stage of the parallel execution of such a B&B method for large scale problems on a large number of SOLVERs, many of the SOLVERs are idle, and it would take a while till all SOLVERs start running. This is clearly inefficient.

If we use branching rule M for a large scale QAP instance, we can start the B&B method at any level of the enumeration tree. For example, consider a QAP instance with $n = 50$. Then we can create $50 \times 49 \times 48 = 117,600$ sub QAPs with dimension 47 at the fourth level in the enumeration tree using branching rules M. We can start the B&B method from these sub QAPs. We will investigate on the performance of this idea.

7.2 Simple estimation of the number of nodes in the enumeration tree

We have proposed 3 types of branching rules M, P and D in Section 5, and observed that the number of nodes generated depends not only on instances but also the branching rule employed. A branching rule sometimes results in much fewer nodes than other rules as observed in bur26a, bur26b and bur26c instances in Table 3. When the B&B method is applied to larger scale QAP instances, selecting a good branching rule becomes a more critical issue. For the selection, we describe a simple sampling method to estimate the number of nodes at each depth (level) of the enumeration tree under a certain uniformity assumption which is necessary for the simplicity of the method and also sufficient for a rough estimation.

Let $G(\mathcal{N}, \mathcal{E})$ denote the enumeration tree to be constructed by applying the B&B method to QAP (2). Recall that each node of $\mathcal{N}$ is identified as a sub QAP, $QAP(F, L)$ for some $(F, L) \in \Pi_r \times \Pi_r$ $(r = 0, 1, \ldots, n)$. The node set $\mathcal{N}$ is subdivided according to the depth of the enumeration tree $G(\mathcal{N}, \mathcal{E})$, depending on their location: $\mathcal{N}_r = \mathcal{N} \cap \big(\Pi_r \times \Pi_r\big)$ $(r = 0, \ldots, n)$. Let $m_r = \mid \mathcal{N}_r \mid$ $(r = 0, 1, \ldots, n)$, which is estimated by the sampling method described below.

Let $r = 0$. Obviously $m_0 = 1$ since $QAP(\emptyset, \emptyset)$ is the unique element of $\mathcal{N}_0$, which corresponds to the root node of the enumeration tree. Hence $\hat{m}_0 = m_0 = 1$. With the application of the lower bounding and upper bounding procedures to $QAP(\emptyset, \emptyset)$, it will become clear whether the B&B method terminates or proceeds to the branching procedure for generating $n - r = n$ child nodes of $QAP(\emptyset, \emptyset)$. In the former case, $\hat{m}_q = m_q = 0$ $(q = 1. \ldots, n)$ and we are done. For the latter case, we have $m_1 = n$ active nodes which forms $\mathcal{N}_1$ and $\hat{m}_1 = m_1 = n$. If the lower bounding and upper bounding procedures are applied to all of the

$\hat{m}_1 = n$ nodes, then the exact m_2 can be obtained; hence $\hat{m}_2 = m_2$. Instead, we choose $0 < s_1 < \hat{m}_1$ nodes randomly from the $\hat{m}_1$ nodes, and apply the procedures to those s_1 nodes to estimate m_2. As a result, some of the s_1 nodes are terminated. Let t_1 denote the number of the nodes that remained active among the s_1 nodes. Applying the branching procedure to those t_1 nodes provides $(n-1)t_1$ nodes (included in $\mathcal{N}_2$) as their child nodes. Thus, each node of the s_1 nodes chosen randomly from the $\hat{m}_1 = m_1$ nodes have generated $(n-1)t_1/s_1$ child nodes on average. Consequently, the number m_2 of the child nodes from the $\hat{m}_1$ nodes is estimated by $\hat{m}_2 = \hat{m}_1 \times (n-1)t_1/s_1$.

Assume that $\hat{m}_q$ $(q = 0, 1, \ldots, r)$ have been computed. We describe how to compute $\hat{m}_{r+1}$ at the rth iteration of the sampling method. If t_{r-1} attained 0, *i.e.*, there are no active nodes from previous iteration, $\hat{m}_r$ was set to zero, so that we set $\hat{m}_{r+1} = 0$. Now we assume on the contrary that $0 < t_{r-1}$ nodes remained active, and $(n-(r-1))t_{r-1}$ nodes included in $\mathcal{N}_r$ were generated in the previous iteration. To compute an estimation $\hat{m}_{r+1}$ of m_{r+1}, choose s_r nodes randomly from the $(n-(r-1))t_{r-1}$ nodes, and apply the lower bounding and upper bounding procedures to them. Suppose that there exist t_r active nodes. Then the branching procedure applied to them yields $(n-r)t_r$ nodes in $\mathcal{N}_{r+1}$. Therefore we set $\hat{m}_{r+1} = \hat{m}_r(n-r)t_r/s_r$.

Table 5: Estimation on the number of nodes in the enumeration tree for QAP instances from QAPLIB. 'Estimated' $= \sum_{r=0}^{n} \hat{m}_r$. 'Generated' $= \sum_{r=0}^{n} m_r$. All $(n-r)!$ feasible solutions of QAP(F, L) were enumerated to compute its exact optimal value when $|F^c| = |L^c| = n - r = 7$ holds; hence $\hat{m}_r = m_r = 0$ for $r = n-6, n-5, \ldots, n$. The **bold blue font** indicates the smallest estimate among the three branching rules M, P and D. - : not computed. (c) — solved at ISM by the C++ code. (m) — solved by the MATLAB code.

	Branching Rule (The number of nodes)					
	M		P		D	
Problem	Estimated	Generated	Estimated	Generated	Estimated	Generated
bur26a	3,038	2,976	15,495	11,401	**2,404**	2,358(m)
bur26b	276,511	144,228	1,106,174	More than 544,719	**7,206**	8,635(m)
bur26c	18,114	16,990	102,053	66,428	**2,609**	2,416(m)
bur26d	119,212	-	7,583,919	-	**5,473**	17,495(m)
bur26e	8,156	-	15,042	-	**1,621**	1,621(m)
bur26f	432,056	-	6,530,633	-	**4,742**	6,536(m)
bur26g	32,935	-	18,023	-	**3,430**	3,265(m)
bur26h	33,963	-	617,498	-	**3,645**	4,211(m)
nug25	3239	8446(c)	**2952**	3004(m)	3,614	3,805(m)
nug27	2660	2610(c)	**746**	979(m)	2,272	2,170(m)
nug28	7,369	11,228(c)	**3,208**	4,236(m)	10,128	9,977(m)
nug30	12,419	26,297(c)	10,905	-	**10,401**	-
tai30a	42,472,865	34,000,579(c)	19,280,014	-	**14,999,704**	-
tai30b	611	989(m)	611	1,654(m)	611	1,150(m)
tai35a	2.1e11	-	3.9e10	-	**1.3e10**	-
tai35b	5,840,218	2,620,547(c)	4,618,822	-	**360,772**	-
tai40b	212,954	278,465(c)	**61,065**	-	71,903,758	-
tai50b	**54,547,664**	-	1.4e9	-	2.1e11	-
tho40	7.1e11	-	**37,382,192**	-	97,834,698	-
sko42	3,575,067	6,019,419(c)	3,071,294	-	**2,764,606**	-
sko49	**1.6e8**	-	1.3e9	-	1.2e9	-
wil50	32,963,810	-	**20,523,849**	-	39,330,160	-

Table 5 shows the estimation of the total number of nodes for some instances. We observe that

(i) (the total number of nodes generated in the numerical experiment)
$\leq 3 \times$ (the estimate of the total number of nodes).

(ii) The branching rule P and D are expected to generate less nodes than the branching rule M except for tai50b and sko49.

(iii) tai50b (using M), tho40 (P) and wil50 (P) could be solved.

(iv) tai35a is too difficult to solve using any of M, P and D.

(v) We may challenge sko49 (M).

References

[1] K. Anstreicher, N. Brixius, Goux J-P., Linderoth, and J. Solving large quadratic assignment problems on computational grids. *Math. Program.*, 91:563–588, 2002.

[2] N. Arima, S. Kim, M. Kojima, and K. C. Toh. Lagrangian-conic relaxations, Part I: A unified framework and its applications to quadratic optimization problems. *Pacific J. Optim.*, 14(1):161–192, 2018.

[3] J. Clausen and M. Perregaard. Solving large quadratic assignment problems in parallel. *Comput. Optim. Appl.*, 8:111–127, 1997.

[4] D. T. Connolly. An improved annealing scheme for the QAP. *Eur. J. Oper. Res.*, 46:93–100, 1990.

[5] L. M. Gambardella, E. D. Taillard, and M. Dorigo. Ant colonies for the QAP. Technical Report IDSIA-4-97, IDSIA, Lugano, Switzerland, 1997.

[6] A. D. Goncalves, A. A. Pessoa, L. M. de A. Drummond, C. Bentes, and R. Farias. Solving the quadratic assignment problem on heterogeneous environment (CPUs and GPUs) with the application of level 2 reformulation and linearization technique. Technical Report arXiv:1510.02065v1, 2015.

[7] P. Hahn and M. Anjos. QAPLIB – A quadratic assignment problem library. http://www.seas.upenn.edu/qaplib.

[8] P. Hahn, A. Roth, and M. Saltzman, M. abd Guignard. Memory-aware parallelized RLT3 for solving quadratic assignment problems. Technical Report Optimization Online, 2013.

[9] N. Ito, S. Kim, M. Kojima, A. Takeda, and K.C. Toh. BBCPOP: A sparse doubly nonnegative relaxation of polynomial optimization problems with binary, box and complementarity constraints. *ACM Trans. Math. Soft.*, 45((3) 34), 2019.

[10] S. Kim, M. Kojima, and K.C. Toh. A Newton-bracketing method for a simple conic optimization problem. *To appear in Optim. Methods and Softw.*

[11] S. Kim, M. Kojima, and K.C. Toh. User manual of newtbracket: "A Newton-Bracketing method for a simple conic optimization problem" with aopplications to QOPs in binary variables. https://sites.google.com/site/masakazukojima1/softwares-developed/newtbracket, November 2020.

[12] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2:83–97, 1955.

[13] D.E. Oliveira, H. Wolkowicz, and Y. Xu. ADMM for the SDP relaxation of the QAP. *Math. Program. Comput.*, 10:631–658, 2018.

[14] P. M. Pardalos, K. G. Ramakrishnan, M. G. C. Resende, and Y. Li. Implementation of a variance reduction-based lower bound in a branch-and-bound algorithm for the quadratic assignment problem. *SIAM J. Optim.*, 7:281–294, 1997.

[15] Ted Ralphs, Yuji Shinano, Timo Berthold, and Thorsten Koch. Parallel solvers for mixed integer linear optimization. In Youssef Hamadi and Lakhdar Sais, editors, *Handbook of Parallel Constraint Reasoning*, pages 283–336. Springer International Publishing, Cham, 2018.

[16] C. Roucairol. A parallel branch and bound algorithm for the quadratic assignment problem. *Discret. Appl. Math.*, 18:211–255, 1987.

[17] H. D. Sherali and C. H. Tuncbilek. A global optimization algorithm for polynomial programming problems using a Reformulation-Linearization Technique. *J. Global Optim.*, 2:101–112, 1992.

[18] Y. Shinano, T. Achterberg, T. Berthold, S. Heinz, and T. Koch. Solving open mip instances with parascip on supercomputers using up to 80,000 cores. 2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pages 770–779. IEEE, 2016.

[19] Y. Shinano, T. Achterberg, T. Berthold, S. Heinz, T. Koch, and M. Winkler. Solving hard MIPLIB2003 problems with ParaSCIP on supercomputers: An update. In *2014 IEEE International Parallel Distributed Processing Symposium Workshops*, pages 1552–1561, 2014.

[20] Yuji Shinano, Tobias Achterberg, Timo Berthold, Stefan Heinz, and Thorsten Koch. Parascip – a parallel extension of scip. In Christian Bischof, Heinz-Gerd Hegering, Wolfgang E. Nagel, and Gabriel Wittum, editors, *Competence in High Performance Computing 2010*, pages 135–148. Springer, February 2012.

[21] Yuji Shinano, Stefan Heinz, Stefan Vigerske, and Michael Winkler. FiberSCIP—a shared memory parallelization of SCIP. *INFORMS Journal on Computing*, 30(1):11–30, 2018.

[22] J. Skorin-Kapov. Tabu search applied to the quadratic assignment problem. *ORSA J. Comput.*, 2:33–45, 1990.

[23] E. Tailard. Robust taboo search for the quadratic assignment problem. *Parallel Comput.*, 17:443–455, 1991.

[24] N. Tateiwa, Y. Shinano, S. Nakamura, A. Yoshida, M. Yasuda, S. Kaji, and K. Fujisawa. Massive parallelization for finding shortest lattice vectors based on ubiquity generator framework. In *2020 SC20: International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 834–848, Los Alamitos, CA, USA, nov 2020. IEEE Computer Society.

[25] L. Q. Yang, D. F. Sun, and K. C. Toh. SDPNAL+: A majorized semismooth Newton-CG augmented Lagrangian method for semidefinite programming with nonnegative constraints. *Math. Prog. Comp.*, 7:331–366, 2015.