MATHEMATICS OF OPERATIONS RESEARCH
Vol. 00, No. 0, Xxxxxx 20xx, pp. xxx–xxx
ISSN 0364-765X | EISSN 1526-5471 |xx|0000|0xxx

DOI 10.1287/moor.xxxx.xxxx
©20xx INFORMS

On the Maximum Quadratic Assignment Problem

Viswanath Nagarajan
IBM T.J. Watson Research center, Yorktown Heights, NY 10598.
email: viswanath@us.ibm.com

Maxim Sviridenko
IBM T.J. Watson Research center, Yorktown Heights, NY 10598.
email: sviri@us.ibm.com

Quadratic Assignment is a basic problem in combinatorial optimization, which generalizes several other problems such as Traveling Salesman, Linear Arrangement, Dense k Subgraph, and Clustering with given sizes. The input to the Quadratic Assignment Problem consists of two $n \times n$ symmetric non-negative matrices $W = (w_{i,j})$ and $D = (d_{i,j})$. Given matrices W, D, and a permutation $\pi : [n] \to [n]$, the objective function is $Q(\pi) := \sum_{i,j \in [n], i \neq j} w_{i,j} \cdot d_{\pi(i),\pi(j)}$. In this paper, we study the *Maximum Quadratic Assignment Problem*, where the goal is to find a permutation π that maximizes $Q(\pi)$. We give an $\tilde{O}(\sqrt{n})$ approximation algorithm, which is the first non-trivial approximation guarantee for this problem. The above guarantee also holds when the matrices W, D are asymmetric. An indication of the hardness of Maximum Quadratic Assignment is that it contains as a special case, the *Dense k Subgraph* problem, for which the best known approximation ratio $\approx n^{1/3}$ (Feige et al. [9]).

When one of the matrices W, D satisfies *triangle inequality*, we obtain a $\frac{2e}{e-1} \approx 3.16$ approximation algorithm. This improves over the previously best-known approximation guarantee of 4 (Arkin et al. [4]) for this special case of Maximum Quadratic Assignment.

The performance guarantee for Maximum Quadratic Assignment with triangle inequality can be proved relative to an optimal solution of a natural linear programming relaxation, that has been used earlier in Branch-and-Bound approaches (see eg. Adams and Johnson [1]). It can also be shown that this LP has an integrality gap of $\tilde{\Omega}(\sqrt{n})$ for general Maximum Quadratic Assignment.

Key words: approximation algorithms; linear programming relaxation

MSC2000 Subject Classification: Primary: 90C27, 90C59, 68W25; Secondary: 68W40, 68W20

OR/MS subject classification: Primary: Analysis of Algorithms - Suboptimal Algorithms; Secondary: Networks/Graphs - Heuristics

History: Received: Xxxx xx, xxxx; Revised: Yyyyyy yy, yyyy and Zzzzzz zz, zzzz.

1. Introduction Quadratic assignment is a basic problem in combinatorial optimization, which generalizes several other problems. The input to quadratic assignment consists of two $n \times n$ symmetric non-negative matrices $W = (w_{i,j})$ and $D = (d_{i,j})$. Given matrices W, D, and a permutation $\pi : [n] \to [n]$, the quadratic assignment objective is $Q(\pi) := \sum_{i,j \in [n], i \neq j} w_{i,j} \cdot d_{\pi(i),\pi(j)}$.

There are two variants of the Quadratic Assignment Problem. In the *Minimum Quadratic Assignment* problem, the objective is to find a permutation π that minimizes $Q(\pi)$. In this paper we study the *Maximum Quadratic Assignment* (Max-QAP) problem, where the objective is to find a permutation π that maximizes $Q(\pi)$.

We present approximation algorithms for the maximum quadratic assignment problem. All problems considered in this paper have maximization objectives. Given a maximization problem Π, a polynomial time algorithm is an *α-approximation algorithm* (for some $\alpha \geq 1$) if on every input instance to Π, the algorithm outputs a feasible solution having objective value at least $1/\alpha$ times the optimal [24].

1.1 Our Results We give an $O(\sqrt{n} \log^2 n)$ approximation algorithm for Max-QAP, which is the first non-trivial approximation guarantee for this problem. In fact, this bound also holds when the matrices W and D are *asymmetric*. Using standard scaling arguments, our algorithm reduces Max-

QAP to a special case (called *0-1* Max-QAP) where the matrices have only 0-1 entries; in this case matrices W, D naturally correspond to a pair of undirected graphs. Our main contribution here is an $O(\sqrt{n})$ approximation algorithm for 0-1 Max-QAP. We note that 0-1 Max-QAP itself contains the dense k subgraph problem as a special case. The algorithm for 0-1 Max-QAP involves taking the better of the following two approaches: **(1)** The first algorithm outputs a random permutation on appropriately chosen (equal-sized) dense subgraphs (or submatrices) of W and D. To find these subgraphs, we use a 2-approximation algorithm for *Vertex Cover* in one graph, and an $\frac{n}{k}$-approximation algorithm for *Dense k Subgraph* in the other graph. **(2)** The second algorithm uses local search to obtain a constant factor approximation for a new problem, *Common Star Packing*, which also defines a feasible solution to Max-QAP. These results appear in Section 2

We also consider a special case of the general Max-QAP *with triangle inequality*, where one of the matrices W, D satisfies triangle inequality. For this case, we give a $\frac{2e}{e-1} \approx 3.16$ approximation algorithm, that improves the previously best known ratio of 4 due to Arkin et al. [4]. Our approach here is as follows. We first define an auxiliary problem and show that it is equivalent (up to a factor 2) to Max-QAP with triangle inequality. This auxiliary problem is another special case of Max-QAP, and it also generalizes the *Maximum Vertex Cover* problem [2, 10]. We obtain an $\frac{e}{e-1}$ approximation algorithm for the auxiliary problem, by rounding a natural LP-relaxation for it. These results appear in Section 3.

In Section 4 we note that a natural LP relaxation (c.f. Adams and Johnson [1]) for Max-QAP can be shown to have an $\tilde{\Omega}(\sqrt{n})$ integrality gap. In fact, in the special case of dense k subgraph, this LP has integrality gap $\tilde{\Theta}(\sqrt{n})$. Furthermore, when restricted to Max-QAP with triangle inequality, this LP has integrality gap at most $\frac{2e}{e-1}$.

An indication of the difficulty in approximating Max-QAP is that it contains the well-studied *dense k subgraph* problem as a special case. The best known approximation guarantee for dense k subgraph is $\approx n^{1/3}$ (Feige et al. [9]). This problem is considered to be fairly hard, however the best known hardness of approximation [8, 18] only rules out the existence of a PTAS (under certain complexity theoretic assumptions).

1.2 Related Work Quadratic assignment is an extensively studied combinatorial optimization problem. The book by Cela [7] surveys several bounding techniques, exact algorithms, and polynomially solvable special cases. Surveys on the quadratic assignment problem include Pardalos and Wolkowitz [20], Loilola et al. [19], and Burkard et al. [6].

Approximation algorithms for maximum quadratic assignment have been obtained in many special cases. One that is most relevant to this paper is a 4-approximation algorithm for Max-QAP when either W or D satisfies the *triangle inequality*, due to Arkin et al. [4]. Another closely related special case is the *dense k subgraph* problem, where W represents an undirected graph and D corresponds to a k-clique. The best known approximation ratio for general dense k subgraph problem is n^c, where $c < \frac{1}{3}$ is some universal constant, due to Feige et al. [9] while the problem where the edge weights satisfy the triangle inequality admits a 2-approximation algorithm due to Hassin et al. [17].

We now list some other special cases of Max-QAP for which approximation algorithms have been considered. In *capacitated star packing* [15, 3], D consists of a set of vertex disjoint stars, and a 3-approximation algorithm is given in Arkin et al. [3]. In obtaining our algorithm for 0-1 Max-QAP, we use a variant (called Common Star Packing) of the capacitated star packing problem, for which we provide a constant approximation algorithm. *Maximum clustering with given sizes* is the special case of Max-QAP when D is the union of vertex disjoint cliques: assuming that W satisfies triangle inequality, Hassin and Rubinstein [16] gave a $(0.5 - 3/k)^{-1}$-approximation algorithm where k is the smallest cluster size. For maximum clustering under a general W matrix, Feo and Khellaf [11] gave an s-approximation when each clique has size s.

For *dense instances* of the 0-1 Max-QAP problem, there is a PTAS known due to Arora et al. [5]. Dense instances are those where both underlying graphs have $\Omega(n^2)$ edges. In our algorithm for Max-QAP with triangle inequality, we encounter a generalization of a previously studied problem 'Maximum Vertex Cover'. The Max-Vertex-Cover problem is APX-hard, and the best known approximation ratio is $\frac{4}{3} - \epsilon$ for some universal constant $\epsilon > 0$, due to Feige and Langberg [10]. Approximation algorithms for maximum bisection problems such as Max-Cut, Dense k subgraph, Max Vertex Cover etc. has been

a very active area of research.

Unlike Max-QAP, the *Minimum Quadratic Assignment* problem remains hard to approximate even when one of the matrices satisfies triangle inequality. Sahni and Gonzales [23] showed that the general case of this problem is hard to approximate to any factor. Queranne [22] showed that it is NP-hard to approximate this problem to any polynomial factor, even when D corresponds to a line metric. Special cases of minimum quadratic assignment, where D is a metric and W corresponds to certain classes of graphs have been studied in [12, 13, 14].

2. General Maximum Quadratic Assignment The *maximum quadratic assignment* (Max-QAP) problem is the following: given two $n \times n$ symmetric non-negative matrices $W = (w_{i,j})$ and $D = (d_{i,j})$, find a permutation π of $[n]$ that maximizes:

$$\sum_{i,j \in [n], i \neq j} w_{i,j} \cdot d_{\pi(i),\pi(j)}$$

We obtain an $O(\sqrt{n}\log^2 n)$ approximation algorithm for this problem. A special case of Max-QAP arises when the matrices W and D have only 0-1 entries, we refer to this problem as *0-1* Max-QAP. At the loss of an $O(\log^2 n)$ factor, we first reduce the general Max-QAP to 0-1 Max-QAP: this step uses standard scaling arguments (Lemma 2.1). Then we obtain an $O(\sqrt{n})$ approximation algorithm for 0-1 Max-QAP (Section 2.1).

LEMMA 2.1 (REDUCTION TO 0-1 Max-QAP) *An α approximation algorithm for 0-1* Max-QAP *implies an $O(\alpha \cdot \log^2 n)$ approximation algorithm for general* Max-QAP.

PROOF. We assume that neither matrix consists of all zeroes in non-diagonal entries, otherwise the problem is trivial. By scaling matrices W and D, we assume that the maximum entry in both matrices is exactly 1; note that the approximation guarantee is invariant under scaling the input matrices. Let Opt denote the optimal value of this Max-QAP instance and π the permutation that achieves this; note that $1 \leq \mathsf{Opt} \leq n^2$. We now modify the matrices W and D, by setting to 0 all entries of value smaller than or equal to $\frac{1}{2n^2}$. This reduces the optimal value by at most $\frac{1}{2} \leq \frac{\mathsf{Opt}}{2}$, so the optimal value of the modified instance is at least $\frac{\mathsf{Opt}}{2}$.

Now partition the entries of matrix W into $g = \lceil \log_2(2n^2) \rceil$ groups so that all entries in the k-th group lie in the interval $(\frac{1}{2^k}, \frac{1}{2^{k-1}}]$. Let A_k denote the $n \times n$ 0-1 matrix that has 1s at all positions corresponding to group k entries and 0s everywhere else. Then we have $\frac{1}{2}W \leq \sum_{k=1}^{g} \frac{1}{2^k} A_k \leq W$. By performing an identical operation on D, we can obtain 0-1 matrices $\{B_k\}_{k=1}^{g}$ such that $\frac{1}{2}D \leq \sum_{k=1}^{g} \frac{1}{2^k} B_k \leq D$. For the optimal permutation π, we can express the objective value corresponding to π as:

$$\begin{aligned}
\sum_{i,j \in [n], i \neq j} w_{i,j} \cdot d_{\pi(i),\pi(j)} &\leq 4 \cdot \sum_{i,j \in [n], i \neq j} \left(\sum_{k=1}^{g} \frac{1}{2^k} A_k(i,j) \right) \cdot \left(\sum_{l=1}^{g} \frac{1}{2^l} B_l(\pi(i),\pi(j)) \right) \\
&= 4 \sum_{k=1}^{g} \sum_{l=1}^{g} \left[\frac{1}{2^{k+l}} \sum_{i,j \in [n], i \neq j} A_k(i,j) \cdot B_l(\pi(i),\pi(j)) \right]
\end{aligned}$$

The left-hand-side above is at least $\frac{\mathsf{Opt}}{2}$, which implies that there exists some pair of values $k, l \in \{1, \cdots, g\}$ such that:

$$\frac{1}{2^{k+l}} \sum_{i,j \in [n], i \neq j} A_k(i,j) \cdot B_l(\pi(i),\pi(j)) \geq \frac{\mathsf{Opt}}{8g^2}$$

Thus if we could approximate 0-1 Max-QAP within an α factor, applying this algorithm to the pair $\langle A_k, B_l \rangle$ gives a permutation σ where,

$$\frac{\mathsf{Opt}}{8g^2\alpha} \leq \frac{1}{2^{k+l}} \sum_{i,j \in [n], i \neq j} A_k(i,j) \cdot B_l(\sigma(i),\sigma(j)) \leq \sum_{i,j \in [n], i \neq j} w(i,j) \cdot d(\sigma(i),\sigma(j)).$$

The last inequality above uses the facts $w(i,j) \geq \frac{1}{2^k} \cdot A_k(i,j)$ and $d(i,j) \geq \frac{1}{2^l} \cdot B_l(i,j)$ for all $i, j \in [n]$.

The algorithm for Max-QAP runs the α-approximation algorithm for 0-1 Max-QAP on all pairs $\langle A_k, B_l \rangle$ (for $1 \leq k, l \leq g$) and returns the best permutation found. From the above, it follows that this is an $O(\alpha \log^2 n)$ approximation algorithm for general Max-QAP. □

2.1 Algorithm for 0-1 Max-QAP In this section, we focus on 0-1 Max-QAP and obtain an $O(\sqrt{n})$ approximation algorithm. In this case, the problem can be stated in terms of two n-vertex undirected simple graphs G and H, where the goal is to find a one-to-one mapping of vertices of G to those of H such that the number of common edges is maximized. For an undirected graph G', we let $E(G')$ denote its set of edges. For graph G' on vertex-set $[n]$ and permutation $\pi : [n] \to [n]$, let $\pi(G')$ denote the graph on vertices $[n]$ with edge-set $E(\pi(G')) = \{(i,j) \mid i,j \in [n], (\pi^{-1}(i), \pi^{-1}(j)) \in E(G')\}$. For two undirected graphs G_1 and G_2 both defined on vertex set $[n]$, $G_1 \cap G_2$ denotes the graph on vertices $[n]$ with $E(G_1 \cap G_2) = E(G_1) \cap E(G_2)$. In graph terms, the *0-1* Max-QAP problem is defined as follows: given undirected graphs G and H each defined on vertex-set $[n]$, find a permutation $\pi : [n] \to [n]$ that maximizes $|E(\pi(G) \cap H)|$.

For the sake of analysis, let π^* denote the optimal permutation and $O = \pi^*(G) \cap H$ the optimal graph, with Opt $= |E(O)|$ edges. It is clear that Opt $\leq \min\{|E(G)|, |E(H)|\}$. Also let k denote the number of *non-isolated* vertices in the optimal graph O; vertex $u \in O$ is non-isolated iff it has degree at least one in O. The final algorithm for 0-1 Max-QAP is the better of two algorithms that we describe next.

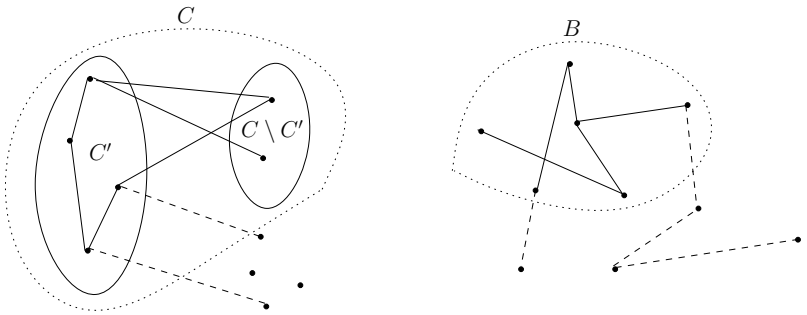
Algorithm 1 (Star packing). Before we present the algorithm, we need some definitions. A *star* in graph G' is a subset of edges $S \subseteq E(G')$ that are all incident to some common vertex (called the *center*). The size of a star is the number of edges in it. A *star packing* in an undirected graph is a collection of vertex-disjoint (non-empty) stars. The *size vector* of a star packing is a tuple $\langle c_1, \cdots, c_p \rangle$ where p denotes the number of stars and $c_1, \cdots, c_p$ denote the sizes of all stars in this packing. Given two undirected graphs G and H, a *common star packing* consists of star packings $\mathcal{S}$ in G and $\mathcal{T}$ in H such that $\mathcal{S}$ and $\mathcal{T}$ have identical size-vectors. The *value* of a common star packing given by a pair $(\mathcal{S}, \mathcal{T})$ of star packings is $\sum_{i=1}^{p} c_i$ where $\langle c_1, \cdots, c_p \rangle$ denotes the common size vector of $\mathcal{S}$ and $\mathcal{T}$. In Section 2.2, we give a 5-approximation algorithm for computing a maximum value common star packing.

The first algorithm for 0-1 Max-QAP involves computing an approximately maximum value common star packing in G and H, using the algorithm in Section 2.2. Observe that any common star packing in graphs G and H of value v naturally corresponds to a solution to 0-1 Max-QAP on G, H with v edges: map corresponding stars in the common star packing to each other.

PROPOSITION 2.1 *The solution computed by Algorithm 1 has $\Omega(k)$ edges.*

PROOF. Consider the optimal graph O and let F denote any spanning forest in O. Since there are k non-isolated vertices in O, forest F has at least $k/2$ edges. For each tree T in forest F, pick an arbitrary root vertex r and assign a level to each edge $e \in T$, which equals the number of edges on the path from e to r in tree T. Observe that we have two star packings: $\mathcal{S}_e$ consisting of all edges in even levels of trees in F, and $\mathcal{S}_o$ consisting of all edges in odd levels. It is easy to verify that $\mathcal{S}_e$ and $\mathcal{S}_o$ are indeed star packings, and that one of them has at least half the edges in F. Thus we have a star packing in O with at least $\frac{k}{4}$ edges. Finally note that any star packing in O corresponds to a common star packing in G and H. Thus running a 5-approximation algorithm for maximum value common star packing gives a solution with at least $\frac{k}{20}$ edges. □

Algorithm 2 (Modified random). The second algorithm involves computing a random mapping between appropriate dense subgraphs of G and H. Recall that the optimal graph O has $n - k$ isolated vertices (i.e. vertices of zero degree). Let $V_1, V_2 \subseteq [n]$ respectively denote the set of G-vertices and H-vertices that are mapped to the isolated vertices in O. For an undirected graph G' on vertex-set $[n]$ and subset $U \subseteq [n]$, $G'[U]$ denotes the subgraph of G' induced on U. Observe that either $E(G[V_1]) = \emptyset$ or $E(H[V_2]) = \emptyset$: otherwise, modifying permutation π^* on vertices V_1 would give a solution with more than Opt edges. Suppose that $E(G[V_1]) = \emptyset$ (the case $E(H[V_2]) = \emptyset$ is identical), then graph G has a vertex cover of size k (namely $[n] \setminus V_1$). In this case, we run a 2-approximation algorithm for vertex-cover on G that computes a set $C' \subseteq [n]$ of $2k$ vertices that covers all edges (c.f. Vazirani [24]). Augment the set C' by adding to it k highest degree vertices from $[n] \setminus C'$ to obtain a set C having $3k$ vertices.



Graph G and its subgraph C Graph H and its subgraph B

Figure 1: The subgraphs used in random mapping.

PROPOSITION 2.2 $|E(G[C])| \geq \mathsf{Opt}$.

PROOF. Since C' is a vertex cover for G, edges of O can be partitioned into: (1) $E_1 \subseteq E(O)$ edges induced on C', and (2) $E_2 \subseteq E(O)$ edges between C' and $[n] \setminus C'$. By definition, $E_1 \subseteq E(G[C']) \subseteq E(G[C])$. Since all edges of O are induced on k vertices, $|E_2|$ is at most the number of edges incident to the k highest degree vertices in $[n] \setminus C'$ (each of which has its other end-point in C'). Thus the number of edges between C' and $C \setminus C'$ is at least $|E_2|$ and $|E(G[C'])| \geq |E_1|$, which implies $|E(G[C])| \geq |E_1| + |E_2| = \mathsf{Opt}$. □

Next we apply an $O(\frac{n}{k})$-approximation algorithm (c.f. Feige et al. [9]) to compute a $3k$-vertex subgraph in H having the maximum number of edges. Let this solution be induced on vertex set B. Note that H contains a k-vertex subgraph with at least Opt edges (corresponding to O), so $H[B]$ contains $\Omega(\frac{k}{n}) \cdot \mathsf{Opt}$ edges. Figure 1 depicts the dense subgraphs in G and H. Algorithm 2 finally returns a *uniformly random* mapping from C to B (other vertices are mapped arbitrarily). Observe that the expected number of common edges in such a random mapping is at least:

$$\frac{1}{(3k)^2} \cdot |E(G[C])| \cdot |E(H[B])| = \frac{1}{(3k)^2} \cdot \Omega\left(\frac{k}{n}\right) \cdot \mathsf{Opt}^2 = \Omega\left(\frac{1}{nk}\right) \cdot \mathsf{Opt}^2$$

since $G[C]$ has $\Omega(\mathsf{Opt})$ edges and $H[B]$ has $\Omega(\frac{k}{n}) \cdot \mathsf{Opt}$ edges.

Finally we output the better of the solutions from Algorithms 1 and 2. The number of edges in this solution is $\max\{\Omega(k), \Omega(\frac{1}{nk})\mathsf{Opt}^2\} \geq \sqrt{\Omega(k \cdot \frac{1}{nk} \cdot \mathsf{Opt}^2)} = \Omega(\frac{1}{\sqrt{n}}) \cdot \mathsf{Opt}$. We note that the second algorithm can be easily derandomized using conditional expectation, to give the following.

THEOREM 2.1 *There is an $O(\sqrt{n})$-approximation algorithm for 0-1* Max-QAP*. Hence there is an $O(\sqrt{n}\log^2 n)$-approximation algorithm for* Max-QAP*.*

Remarks. A possible simplification to our algorithm could be just to output the better of maximum common star packing and a uniformly random permutation. However this algorithm achieves only an approximation ratio $\Omega(n^{2/3})$ as shown by an example where both graphs G and H are cliques on $n^{2/3}$ vertices. We note that this simpler algorithm can in fact be shown to achieve a $\Theta(n^{2/3})$ approximation guarantee. Hence in our algorithm, it is important to find appropriate dense subgraphs before applying a random permutation. A tight example for our algorithm is when G and H are identical $\sqrt{n}$ regular graphs containing a perfect matching: both Algorithms 1 and 2 return solutions of value $O(n)$, whereas the optimal value is $\Omega(n\sqrt{n})$.

2.2 Maximum Value Common Star Packing In this section, we consider the maximum value common star packing problem: given two undirected n-vertex graphs G, H and a number $1 \leq p \leq n$, compute a maximum value common star packing in G and H that consists of exactly p stars. We present a 5-approximation algorithm for this problem. A related problem is *capacitated star packing* [3], where given a single weighted complete graph and a fixed size vector $\overline{s} = \langle s_1, \cdots, s_p \rangle$, the goal is to compute a maximum weight star packing having size vector $\overline{s}$. Arkin et al. [3] gave a 3-approximation algorithm for

capacitated star packing. Our algorithm for *common star packing* is based on local-search and is similar to the algorithm in [3].

In Algorithm 1 for 0-1 Max-QAP, we require the maximum value common star packing when the number of stars p is not fixed: for this purpose we run the algorithm for fixed p (described below) over all values of $1 \leq p \leq n$, and pick the best common star packing.

The algorithm for common star packing always maintains a common star packing given by a pair of star packings $\mathcal{S} = \{S_1, \cdots, S_p\}$ in G and $\mathcal{T} = \{T_1, \cdots, T_p\}$ in H, where S_i and T_i have the same size (for all $1 \leq i \leq p$). For a common star packing $\langle \mathcal{S}, \mathcal{T} \rangle$ as above, we denote by $E(\mathcal{S}) = \bigcup_{i=1}^{p} E(S_i)$ (resp. $E(\mathcal{T}) = \bigcup_{i=1}^{p} E(T_i)$) the set of edges in $\mathcal{S}$ (resp. $\mathcal{T}$). Observe that the value of this common star packing is $|E(\mathcal{S})| = |E(\mathcal{T})|$. We define a bijection $\sigma : E(\mathcal{S}) \rightarrow E(\mathcal{T})$ that maps each edge in $E(S_i)$ to some edge in $E(T_i)$ (for every $1 \leq i \leq p$). Given common star packing $\langle \mathcal{S}, \mathcal{T} \rangle$, a local move is specified by a tuple $\langle i, x_i, y_i, c_i' \rangle$ where:

- Index $1 \leq i \leq p$ specifies a pair of stars $S_i \in \mathcal{S}$ and $T_i \in \mathcal{T}$.
- $x_i \in G$ and $y_i \in H$ denote new centers of the i^{th} stars.
- $0 \leq c_i' \leq n$ denotes the new size of the i^{th} stars.

Let $v = |E(\mathcal{S})| = |E(\mathcal{T})|$ denote the value of the common star packing. Applying move $\langle i, x_i, y_i, c_i' \rangle$ to $\langle \mathcal{S}, \mathcal{T} \rangle$ involves the following modifications (below, two edges are called independent if they are not incident to a common vertex).

(i) Remove edges $E(S_i)$ from $E(\mathcal{S})$ and $E(T_i) = \sigma(E(S_i))$ from $E(\mathcal{T})$.

(ii) Let $X_i \subseteq E(\mathcal{S})$ denote the edges of $E(\mathcal{S})$ incident to vertex x_i. Remove X_i from $E(\mathcal{S})$ and $\sigma(X_i)$ from $E(\mathcal{T})$.

(iii) Let $Y_i \subseteq E(\mathcal{T})$ denote the edges of $E(\mathcal{T})$ incident to vertex y_i. Remove Y_i from $E(\mathcal{T})$ and $\sigma^{-1}(Y_i)$ from $E(\mathcal{S})$.

(iv) Let A_i denote a set of c_i' edges incident to vertex x_i in G such that each edge in A_i is independent of all edges in $E(\mathcal{S})$. If there does not exist such an A_i, the local move *fails*.

(v) Let B_i denote a set of c_i' edges incident to vertex y_i in H such that each edge in B_i is independent of all edges in $E(\mathcal{T})$. If there does not exist such a B_i, the local move *fails*.

(vi) Add A_i to $E(\mathcal{S})$ and B_i to $E(\mathcal{T})$, and augment bijection σ so that $\sigma(A_i) = B_i$.

In steps (i)-(iii), we only remove corresponding pairs of edges (under bijection σ) from $E(\mathcal{S})$ and $E(\mathcal{T})$. This ensures that after these modifications $\langle \mathcal{S}, \mathcal{T} \rangle$ remains a feasible common star packing. Furthermore, the value of $\langle \mathcal{S}, \mathcal{T} \rangle$ after step (iii) is $v - |S_i| - |X_i| - |Y_i|$. If the local move does not fail, then we obtain sets A_i and B_i in steps (iv)-(v). Note that A_i (resp. B_i) corresponds to a c_i' size star centered at x_i (resp. y_i) in graph G (resp. H). By its definition, star A_i (resp. B_i) can be added to $\mathcal{S}$ (resp. $\mathcal{T}$) to obtain a star packing. Since $|A_i| = |B_i| = c_i'$, after step (vi) $\langle \mathcal{S}, \mathcal{T} \rangle$ is a common star packing of value $v + c_i' - |S_i| - |X_i| - |Y_i|$. Finally the local move is said to be *improving* iff it does not fail and the increase in value $c_i' - |S_i| - |X_i| - |Y_i| > 0$.

The algorithm is initialized with $\mathcal{S}$ and $\mathcal{T}$ being zero-value star packings in graphs G and H respectively. Then it performs any sequence of improving local moves, until no further improvement is possible. The value of the solution increases by at least one in each step, and the maximum value of a common star packing is n (number of vertices). So the number of iterations is at most n. The number of local moves at any step is at most n^4, and each local move (steps (i)-(vi)) can be easily performed in polynomial time. Thus the entire algorithm runs in polynomial time.

We now argue that any locally optimal solution is a 5-approximate solution. Let $\mathcal{S} = \{S_1, \cdots, S_p\}$ in G and $\mathcal{T} = \{T_1, \cdots, T_p\}$ in H denote the common star packing at a local optimum, where $|S_i| = |T_i| = c_i$ for all $1 \leq i \leq p$. Similarly let $\mathcal{S}^* = \{S_1^*, \cdots, S_p^*\}$ in G and $\mathcal{T}^* = \{T_1^*, \cdots, T_p^*\}$ in H denote the optimal common star packing, where $|S_i^*| = |T_i^*| = c_i^*$ for all $1 \leq i \leq p$. Define $\mathsf{touch}(S_i^*)$ to be the number of edges in $\mathcal{S}$ that have a vertex in common with star S_i^*, and $\mathsf{touch}(T_i^*)$ the number of edges in $\mathcal{T}$ having a vertex in common with star T_i^*. Since $\mathcal{S}$, $\mathcal{S}^*$, $\mathcal{T}$, and $\mathcal{T}^*$ are star packings, $\sum_{i=1}^{p} \mathsf{touch}(S_i^*) \leq 2 \sum_{i=1}^{p} c_i$ and $\sum_{i=1}^{p} \mathsf{touch}(T_i^*) \leq 2 \sum_{i=1}^{p} c_i$.

PROPOSITION 2.3 *For any* $1 \leq i \leq p$, $c_i^* - c_i - \mathsf{touch}(S_i^*) - \mathsf{touch}(T_i^*) \leq 0$.

PROOF. Fix a $1 \leq i \leq p$. Suppose for a contradiction that $c_i^* - c_i - \mathsf{touch}(S_i^*) - \mathsf{touch}(T_i^*) > 0$. Let $x_i \in G$ be the center of star S_i^* and $y_i \in H$ be the center of star T_i^*. Let α_i (resp. β_i) denote the number of edges in $\mathcal{S}$ (resp. $\mathcal{T}$) incident to x_i (resp. y_i). Define $c_i' := c_i^* - \mathsf{touch}(S_i^*) - \mathsf{touch}(T_i^*) + \alpha_i + \beta_i$. Consider the local move $\langle i, x_i, y_i, c_i' \rangle$. Observe that when this move is applied to $\langle \mathcal{S}, \mathcal{T} \rangle$, we have $|X_i| = \alpha_i$ and $|Y_i| \leq \beta_i$ in steps (ii) and (iii) respectively. Since $\mathsf{touch}(S_i^*)$ is the number of edges of $\mathcal{S}$ incident to star S_i^* and $|X_i| = \alpha_i$ is the number of edges in $\mathcal{S}$ incident to x_i, there are at least $|S_i^*| - (\mathsf{touch}(S_i^*) - \alpha_i)$ edges of star S_i^* that are independent of $E(\mathcal{S})$ in step (iv). Now observe that $c_i' \leq c_i^* + \alpha_i - \mathsf{touch}(S_i^*)$, and hence step (iv) succeeds. By an identical argument, it follows that step (v) also succeeds. Finally, the increase in value by this local move is:

$$c_i' - c_i - |X_i| - |Y_i| \geq c_i^* - c_i - \mathsf{touch}(S_i^*) - \mathsf{touch}(T_i^*) > 0$$

But this contradicts the fact that $\langle \mathcal{S}, \mathcal{T} \rangle$ is a local optimum. Thus we have the claim. □

Adding the p expressions given by Proposition 2.3, we have:

$$\sum_{i=1}^{p} c_i^* - \sum_{i=1}^{p} c_i - \sum_{i=1}^{p} \mathsf{touch}(S_i^*) - \sum_{i=1}^{p} \mathsf{touch}(T_i^*) \leq 0 \Rightarrow \sum_{i=1}^{p} c_i^* - 5 \sum_{i=1}^{p} c_i \leq 0,$$

since $\sum_{i=1}^{p} \mathsf{touch}(S_i^*) \leq 2 \sum_{i=1}^{p} c_i$ and $\sum_{i=1}^{p} \mathsf{touch}(T_i^*) \leq 2 \sum_{i=1}^{p} c_i$. Hence any local optimum is a 5-approximation and we obtain the following theorem.

THEOREM 2.2 *There is a 5-approximation algorithm for maximum value common star packing.*

2.3 Asymmetric Maximum Quadratic Assignment We note that our algorithm for the general Max-QAP problem extends readily to the case when the matrices W, D are asymmetric. The reduction to 0-1 Max-QAP (Lemma 2.1) clearly holds in the asymmetric case as well. Hence it suffices to consider the directed version of 0-1 Max-QAP, where given two n-vertex directed graphs, the goal is to find a permutation of one graph that maximizes the number of common edges. Following the notation in Section 2.1, if k denotes the number of non-isolated vertices in the optimal graph O, then Claim 2.1 implies that O contains a star-packing (in the undirected sense) of size at least $k/4$. It follows that there is either an In-star packing (where edges of each star are directed to its center) or an Out-star packing (where edges of each star are directed away from its center) having size $k/8$. The Common Star Packing algorithm of Section 2.2 easily extends to give a constant factor approximation for computing a maximum value common In-star (resp. Out-star) packing in two directed graphs. So Algorithm 1 is guaranteed to find a solution of value $\Omega(k)$.

In Algorithm 2, we consider both graphs as being undirected. Then exactly as before, we obtain two $3k$ vertex subgraphs such that one of them has $\Omega(1) \cdot \mathsf{Opt}$ edges and the other $\Omega\left(\frac{k}{n}\right) \cdot \mathsf{Opt}$ edges. Finally observe that a uniformly random mapping of two r-vertex directed graphs having m_1 and m_2 edges results in at least $\frac{m_1 \cdot m_2}{r^2}$ common *directed* edges in expectation. So Algorithm 2 outputs a solution of value $\Omega\left(\frac{1}{nk}\right) \cdot \mathsf{Opt}^2$. Thus we obtain the following.

COROLLARY 2.1 *There is an $O(\sqrt{n} \log^2 n)$ approximation algorithm for asymmetric* Max-QAP.

3. Maximum Quadratic Assignment Problem with Triangle Inequality In this section we treat the special case of the Maximum Quadratic Assignment Problem where the elements of matrix D satisfy the triangle inequality, i.e. $d_{ij} + d_{jk} \geq d_{ik}$ for all $i, j, k \in [n]$. In this case we give an improved $\frac{2e}{e-1}$-approximation algorithm.

Let G and H be the complete undirected graphs with edge weights defined by matrices W and D, respectively. Let M be the matching in graph H obtained by the straightforward greedy algorithm: pick the heaviest edge in graph H and delete all incident edges, in the remaining graph choose the heaviest edge and so on. We will call such a matching M *greedy*. Note that $|M| = \lfloor n/2 \rfloor$ since H is a complete graph. Each edge e of graph H is incident with either one or two edges of matching M. For any edge $e \in H$, let $m(e)$ be the edge in M with the largest weight that is incident to e. By the construction of the greedy matching M, we have $d_{m(e)} \geq d_e, \ \forall e \in H$.

We consider the following modification of the given Max-QAP instance, which we call the *auxiliary problem*. Find a permutation π of $[n]$ that maximizes:

$$\sum_{i,j\in[n],i\neq j} w_{i,j}\cdot d_{m(\pi(i),\pi(j))}, \tag{1}$$

i.e. the weight of each edge (i,j) in graph G is multiplied by the weight of edge $m(\pi(i),\pi(j))$ in graph H (which is incident to the edge $(\pi(i),\pi(j))$ in H). Let Opt^* be the value of the optimal solution to the Max-QAP instance, and let Aux^* be the optimal value of the auxiliary problem (1). We first prove a simple lemma based on triangle inequality.

LEMMA 3.1 $\mathsf{Aux}^* \geq \mathsf{Opt}^* \geq \mathsf{Aux}^*/2$

PROOF. Since $d_{m(e)} \geq d_e$ for all edges $e \in H$, we obtain the first inequality $\mathsf{Aux}^* \geq \mathsf{Opt}^*$.

Consider now an optimal solution (permutation) σ for the auxiliary problem (1), that maps vertices of G to those of H. Let σ' denote the random permutation where we swap assignments along each edge of matching M with probability $1/2$. More precisely, consider an edge $(u,v) \in M$ such that i and j are the two vertices of graph G mapped to the endpoints of this, i.e. $u = \sigma(i)$ and $v = \sigma(j)$. We set $\sigma'(i) = u$, $\sigma'(j) = v$ with probability $1/2$, and $\sigma'(i) = v$, $\sigma'(j) = u$ with probability $1/2$. This process is repeated independently for all edges of the greedy matching M.

We now prove that the expected value of the original Max-QAP instance on the random permutation σ' is at least $\mathsf{Aux}^*/2$ that would imply the second inequality of the lemma. Consider an edge (i,j) in graph G, and let $\sigma(i) = u$ and $\sigma(j) = v$. If $(u,v) \in M$, then the expectation of the term corresponding to (i,j) in the objective function of Max-QAP on permutation σ' is exactly $w_{ij}d_{u,v}$. If $(u,v) \notin M$ and both u and v are incident to some edge from M, then let $\bar{u}$ and $\bar{v}$ be the other endpoints of edges from M incident to u and v, i.e. $(u,\bar{u}) \in M$ and $(v,\bar{v}) \in M$. In this case, by triangle inequality the expectation of the objective function term corresponding to (i,j) on permutation σ' is exactly:

$$w_{ij}\cdot\left(\frac{d_{u,v}+d_{u,\bar{v}}+d_{\bar{u},v}+d_{\bar{u},\bar{v}}}{4}\right) \;\geq\; w_{ij}\cdot\frac{\max\{d_{u,\bar{u}},d_{v,\bar{v}}\}}{2} \;=\; w_{ij}\cdot\frac{d_{m(\sigma(i),\sigma(j))}}{2},$$

where the first inequality uses triangle inequality on d. Analogously, if vertex u or v (say v) is the single vertex of H that is not incident to any edge of the greedy matching M and $(u,\bar{u}) \in M$, then the expectation of the objective function term corresponding to (i,j) on permutation σ' is (again using triangle inequality on d):

$$w_{ij}\cdot\left(\frac{d_{u,v}+d_{\bar{u},v}}{2}\right) \;\geq\; w_{ij}\cdot\frac{d_{u,\bar{u}}}{2} \;=\; w_{ij}\cdot\frac{d_{m(\sigma(i),\sigma(j))}}{2}.$$

Summing up the contribution to the Max-QAP objective over all edges $(i,j) \in G$, the expected value of permutation σ' is at least $\frac{\mathsf{Aux}^*}{2}$ which implies $\mathsf{Opt}^* \geq \mathsf{Aux}^*/2$. □

3.1 Algorithm for the auxiliary problem In the rest of the section, we will show how to construct a $(1-\frac{1}{e})$ approximation algorithm for the auxiliary problem. We consider the following more general problem. The input is an undirected edge-weighted graph $G = (V,E,w)$ with nonnegative edge weights $w_e \geq 0$ for $e \in E$ and nonnegative numbers $\{\Delta_i\}_{i=1}^n$. The goal is to find a permutation π of vertices of graph G that maximizes

$$2\cdot\sum_{(i,j)\in E} w_{ij}\cdot\left(\sum_{p=\min\{\pi_i,\pi_j\}}^{n}\Delta_p\right) \tag{2}$$

We first reduce the auxiliary problem (1) to one of the above form (2). The weighted graph G is the complete graph on vertex set $V = [n]$ with edge-weights W. Let $l = \lfloor n/2 \rfloor$ and $D_1 \geq D_2 \geq \cdots \geq D_l$ be the edge-weights of greedy matching M. We set $\Delta_{2q} = D_q - D_{q+1}$ for all $1 \leq q \leq l$ (here $D_{l+1} = 0$), and all other Δs are set to 0. We also renumber vertices in graph H, in the auxiliary problem (1), so that the edges in the greedy matching M are chosen in the order $(1,2),\ (3,4),\cdots,\ (2l-1,2l)$. Note that for any permutation π and vertices $i,j \in G$, we have

$$d_{m(\pi(i),\pi(j))} = \sum_{p=\min\{\pi(i),\pi(j)\}}\Delta_p$$

Hence for every permutation π, objective (2) is equivalent to objective (1): note that (2) sums over unordered pairs (i,j) whereas (1) sums over ordered pairs, and since W, D are symmetric the two objectives are equal. In the following we obtain an $\frac{e}{e-1}$-approximation algorithm for problem (2).

Problem (2) generalizes the *Maximum Vertex Cover* problem where, given an edge-weighted undirected graph and a number k, the goal is to find k vertices that cover the maximum weight of edges. The maximum vertex cover problem is APX-hard [21] and the best known approximation ratio is $\approx \frac{3}{4}$ [10]. We present a $(1-\frac{1}{e})$ approximation algorithm for problem (2) using a natural LP relaxation. In the following, x-variables are assignment variables mapping vertices to positions, and each variable z_{ijs} denotes whether either of vertices $i, j \in V$ is mapped to some position in $\{1, \cdots, s\}$ (where $z \in [n]$).

$$\max \quad 2 \cdot \sum_{(i,j)\in E} w_{ij} \sum_{s=1}^{n} \Delta_s \cdot z_{ijs}, \tag{3}$$

$$\text{s.t.} \quad z_{ijs} \le \sum_{t=1}^{s} x_{it} + \sum_{t=1}^{s} x_{jt}, \qquad \forall\ (i,j) \in E, \quad \forall\ s = 1,\dots,n \tag{4}$$

$$z_{ijs} \le 1, \qquad \forall\ (i,j) \in E, \quad \forall\ s = 1,\dots,n \tag{5}$$

$$\sum_{i\in V} x_{it} = 1, \qquad \forall\ t = 1,\dots,n \tag{6}$$

$$\sum_{t=1}^{n} x_{it} = 1, \qquad \forall\ i \in V \tag{7}$$

$$x_{it} \ge 0, \qquad \forall\ i \in V, \quad \forall\ t = 1,\dots,n \tag{8}$$

$$z_{ijs} \ge 0, \qquad \forall\ (i,j) \in E, \quad \forall\ s = 1,\dots,n \tag{9}$$

Our algorithm is a natural randomized rounding of the optimal solution (x^*, z^*) of the above linear program. For each position $t = 1,\dots,n$ we treat constraint (6) as a density function and choose a vertex $i \in V$ at random according to this distribution, to assign to position t. After that each position $t = 1,\dots,n$ has one chosen vertex. If a vertex is chosen by many positions then we assign it to the *earliest* one. The vertices not chosen by any position in the previous step of the algorithm are assigned to arbitrary empty positions.

We now derive the expected performance guarantee of the algorithm. For each edge $(i,j) \in E$ we estimate its contribution to the objective function:

$$\begin{aligned}
w_{ij} \sum_{s=1}^{n} \Delta_s \cdot Pr(i \text{ or } j \text{ is assigned a position } \le s) &= w_{ij} \sum_{s=1}^{n} \Delta_s \cdot \left(1 - \prod_{t=1}^{s} (1 - x^*_{it} - x^*_{jt})\right) && (10)\\
&\ge w_{ij} \sum_{s=1}^{n} \Delta_s \left(1 - \exp\left(-\sum_{t=1}^{s} (x^*_{it} + x^*_{jt})\right)\right) && (11)\\
&\ge w_{ij} \sum_{s=1}^{n} \Delta_s \left(1 - \frac{1}{e}\right) \cdot \min\left\{\sum_{t=1}^{s} (x^*_{it} + x^*_{jt}), 1\right\} && (12)\\
&= \left(1 - \frac{1}{e}\right) \cdot w_{ij} \sum_{s=1}^{n} \Delta_s z^*_{ijs}
\end{aligned}$$

Inequality (11) follows from the fact that $1 + x \le e^x$ for all $x \in \mathbb{R}$, and inequality (12) from $1 - e^{-x} \ge (1 - \frac{1}{e})x$ for $0 \le x \le 1$. Therefore, the total expected objective function value of the rounded solution is at least $1 - \frac{1}{e}$ times the optimal value of the linear programming relaxation. Combined with Lemma 3.1, we have the following.

THEOREM 3.1 *There is a $\frac{2e}{e-1}$ approximation algorithm for* Max-QAP *with triangle inequality.*

Derandomization. The above randomized rounding algorithm can be derandomized using *conditional expectations* since we have an exact expression for the expected objective value (10). Similarly, the algorithm in Lemma 3.1, that obtains a solution to Max-QAP from one for problem (1) can be derandomized. Hence we obtain a deterministic algorithm in Theorem 3.1.

4. Some Remarks on an LP Relaxation for Maximum Quadratic Assignment Consider the following integer program for Max-QAP. We have assignment variables $x_{i,p}$ corresponding to a mapping between vertices of the two graphs, and variables $y_{i,p,j,q}$ denote whether "i maps to p *and* j maps to q". The LP relaxation $\mathsf{LP_{QAP}}$ is obtained by dropping the integrality condition on variables, and is given below.

$$
\begin{array}{lll}
\max & \sum_{i,j\in[n],i\neq j} w_{ij} \sum_{p,q\in[n],p\neq q} d_{pq}\cdot y_{i,p,j,q}, & \\
\text{s.t.} & \sum_{i=1}^{n} x_{i,p} = 1, & \forall\ 1\le p\le n \\
 & \sum_{p=1}^{n} x_{i,p} = 1, & \forall\ 1\le i\le n \\
 & \sum_{i=1}^{n} y_{i,p,j,q} = x_{j,q}, & \forall\ 1\le p,j,q\le n \\
(\mathsf{LP_{QAP}}) & \sum_{p=1}^{n} y_{i,p,j,q} = x_{j,q}, & \forall\ 1\le i,j,q\le n \\
 & \sum_{j=1}^{n} y_{i,p,j,q} = x_{i,p}, & \forall\ 1\le i,p,q\le n \\
 & \sum_{q=1}^{n} y_{i,p,j,q} = x_{i,p}, & \forall\ 1\le i,j,p\le n \\
 & x_{i,p}\ge 0, & \forall\ 1\le i,p\le n \\
 & y_{i,p,j,q}\ge 0, & \forall\ 1\le i,p,j,q\le n.
\end{array}
$$

General Max-QAP. The *dense k subgraph* problem is the special case of Max-QAP when matrix D is the incidence matrix of a k-clique (i.e. $d_{pq}=1$ if $1\le p,q\le k$, and $d_{pq}=0$ otherwise), and W is the incidence matrix of the input graph. We note that in the case of dense k subgraph, this LP can be shown to have an integrality gap of $O(\sqrt{n})$: $\mathsf{LP_{QAP}}$ is at least as good as the standard LP for dense k subgraph, which has integrality gap $\approx \frac{n}{k}$ (due to Goemans); in addition $\mathsf{LP_{QAP}}$ cannot have value larger than $\min\{k^2, m\}$ (where m is number of edges in the input graph), so its integrality gap is at most k. The following example shows that this is nearly tight. Consider $k=\sqrt{n}$, and a random k-regular graph (see Wormald [25] for such models) corresponding to W. With high probability, the optimal value of the integer program can be bounded by $O(k\log n)$. However, the LP solution when all $x_{i,p}=\frac{1}{n}$ can be shown to have objective value $\Omega(k^2)$. This gives an $\Omega(\frac{\sqrt{n}}{\log n})$ lower bound on the integrality gap of $\mathsf{LP_{QAP}}$.

Triangle inequality Max-QAP. We also observe that for Max-QAP with triangle inequality, our approximation algorithm (Section 3) implies the same upper bound on the integrality gap of the above LP relaxation. Given an optimal solution (x,y) to $\mathsf{LP_{QAP}}$ for Max-QAP, we induce a solution $(\widetilde{x},z)$ for the LP relaxation for problem (2), where $\widetilde{x}_{it}=x_{it}$ for all $i,t\in[n]$ and $z_{ijs}=\sum_{p=1}^{s}(\sum_{q=1}^{n} y_{ipjq}+\sum_{q=s+1}^{n} y_{iqjp})$ for all $1\le i<j\le n$, $s\in[n]$. We claim that $z_{ijs}\le 1$ and $z_{ijs}\le\sum_{t=1}^{s}(\widetilde{x}_{it}+\widetilde{x}_{jt})$. Indeed,

$$z_{ijs} \;=\; \sum_{p=1}^{s}\sum_{q=1}^{n} y_{ipjq} + \sum_{q=s+1}^{n}\sum_{p=1}^{s} y_{iqjp} \;\le\; \sum_{p=1}^{n}\sum_{q=1}^{n} y_{ipjq} \;=\; 1,$$

$$z_{ijs} \;=\; \sum_{p=1}^{s}\sum_{q=1}^{n} y_{ipjq} + \sum_{p=1}^{s}\sum_{q=s+1}^{n} y_{iqjp} \;\le\; \sum_{p=1}^{s}\sum_{q=1}^{n} y_{ipjq} + \sum_{p=1}^{s}\sum_{q=1}^{n} y_{iqjp} \;=\; \sum_{p=1}^{s}(x_{ip}+x_{jp})$$

So $(\widetilde{x},z)$ is indeed feasible to linear program (3)-(9). Now in place of the first inequality in Lemma 3.1,

we can argue that the LP-objective in (3) of $(\widetilde{x}, z)$ is at least the LP-objective of (x, y) in $\mathsf{LP_{QAP}}$. Indeed,

$$\begin{aligned}
w_{ij}\sum_{s=1}^{n}\Delta_s z_{ijs} &= \\
w_{ij}\sum_{s=1}^{n}\Delta_s\sum_{p=1}^{s}(\sum_{q=1}^{n}y_{ipjq}+\sum_{q=s+1}^{n}y_{iqjp}) &= \\
w_{ij}\sum_{p,q\in[n],p\neq q}y_{ipjq}\left(\sum_{s=p}^{n}\Delta_s+\sum_{s=q}^{p-1}\Delta_s\right) &= \\
w_{ij}\sum_{p,q\in[n],p\neq q}y_{ipjq}\sum_{s=\min(p,q)}^{n}\Delta_s &= \\
w_{ij}\sum_{p,q\in[n],p\neq q}d_{m(p,q)}y_{ipjq} &\geq \\
w_{ij}\sum_{p,q\in[n],p\neq q}d_{pq}y_{ipjq}. &
\end{aligned}$$

Then the LP-rounding algorithm for problem (2) together with the second part of Lemma 3.1 implies that the integrality gap of $\mathsf{LP_{QAP}}$ for Max-QAP is at most $\frac{2e}{e-1}$, when one of the matrices satisfies triangle inequality.

References

[1] W. P. Adams and T. A. Johnson, Improved Linear Programming-based Lower Bounds for the Quadratic Assignment Problem, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 16, 1994, 43-77.

[2] A. Ageev and M. Sviridenko, Pipage rounding: a new method of constructing algorithms with proven performance guarantee, J. Comb. Optim. 8 (2004), pp. 307–328.

[3] E. Arkin, R. Hassin, S. Rubinstein and M. Sviridenko, Approximations for Maximum Transportation Problem with Permutable Supply Vector and Others Capacitated Star Packing Problems, Algorithmica, 39, 2004, 175-187.

[4] E.M. Arkin, R. Hassin and M. Sviridenko, Approximating the maximum quadratic assignment problem, Information Processing Letters, 77, 2001, 13-16.

[5] S. Arora, A. Frieze and H. Kaplan, A new rounding procedure for the assignment problem with applications to dense graph arrangement problems, Mathematical Programming, 92(1), 2002, 1-36.

[6] R.E. Burkard, E. Cela, P. Pardalos and L.S. Pitsoulis, The quadratic assignment problem, In Handbook of Combinatorial Optimization, D.Z. Du, P.M. Pardalos (Eds.), Vol. 3, Kluwer Academic Publishers, 1998, 241-339.

[7] Eranda Cela, The Quadratic Assignment Problem: Theory and Algorithms, Springer, 1998.

[8] Uriel Feige, Relations between average case complexity and approximation complexity, In Proceedings of the 34^{th} Annual ACM Symposium on Theory of Computing, 2002, 534-543.

[9] Uriel Feige, Guy Kortsarz, and David Peleg, The Dense k-Subgraph Problem, Algorithmica, 29(3), 410-421, 2001.

[10] Uriel Feige and Michael Langberg, Approximation Algorithms for Maximization Problems Arising in Graph Partitioning, Journal of Algorithms, 41(2), 2001, 174-211.

[11] T. Feo and M. Khellaf, A class of bounded approximation algorithms for graph partitioning, Networks, 20, 1990, 181-195.

[12] N. Guttmann-Beck and R. Hassin, Approximation algorithms for minimum tree tree partition, Discrete Applied Mathematics, 87, 1998, 117-137.

[13] N. Guttmann-Beck and R. Hassin, Approximation algorithms for min-sum p-clustering, Discrete Applied Mathematics, 89, 1998, 125-142.

[14] R. Hassin, A. Levin and M. Sviridenko, Approximating the minimum quadratic assignment problems, Submitted for publication.

[15] R. Hassin and S. Rubinstein, Robust matchings. SIAM J. Discrete Math. 15 (2002), pp. 530–537.

[16] R. Hassin and S. Rubinstein, An improved approximation algorithm m for the metric maximum clustering problem with given cluster sizes, Information Processing Letters 98 (2006), pp. 92-95.

[17] R. Hassin, S. Rubinstein and A. Tamir, Approximation Algorithms for Maximum Dispersion, Operations Research Letters 21 (1997), pp. 133-137.

[18] Subhash Khot, Ruling Out PTAS for Graph Min-Bisection, Dense k-Subgraph, and Bipartite Clique, SIAM J. Comput., 36(4), 2006, 1025-1071.

[19] E.M. Loilola, N.M.M. De Abreu, P.O. Boaventura-Netto, P.M. Hahn, and T. Querido, A survey for the quadratic assignment problem, Invited Review, European Journal of Operational Research, 176, 657-690, 2006.

[20] P. Pardalos and H. Wolkowitz, eds., Proceedings of the DIMACS Workshop on Quadratic Assignment Problems, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 16, 1994.

[21] E. Petrank, The hardness of approximation: gap location, Computational Complexity, 4, 1994, 133-157.

[22] M. Queyranne, Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems, Operations Research Letters, 4, 1986, 231-234.

[23] S. Sahni and T. Gonzalez, P-complete approximation problems, J. ACM, 23, 1976, 555-565.

[24] V. Vazirani, Approximation Algorithms, Springer, 2002.

[25] N.C. Wormald, Models of random regular graphs. Surveys in Combinatorics, 1999, J.D. Lamb and D.A. Preece, eds. London Mathematical Society Lecture Note Series, vol 276, pp. 239-298.