Polynomial Algorithms for Some Quadratic Assignment Problems in Terms of Graphs

G. G. Zabudsky*, A. Yu. Lagzdin†

*Omsk Branch of Sobolev Institute of Mathematics, SB RAS, zabudsky@ofim.oscsbras.ru
†Omsk Branch of Sobolev Institute of Mathematics, SB RAS, art.lagzdin@gmail.com

1 Introduction

The quadratic assignment problem (QAP) is a well-known problem in field of discrete optimization [4]. It formalizes the problem of placing interrelated objects. A lot of real-life applications, such as campus planning, computer motherboard designing, scheduling of parallel production lines and so on, can be modeled as the QAPs [1, 2]. Some opimization problems, for example, the traveling salesman problem and the maximum clique problem, are special cases of the QAP. The QAP is NP-complete in common case.

We notice that for the problem with number of the objects greater than 20 there is no algorithm to find the optimal solution in efficient time. Parallel algorithms implemented on supercomputers and use of grid computing allow to raise the size of such instances up to 25.

We consider the QAP in terms of graphs [2]. A weighted undirected graph represents a structure of links between the objects. This graph is called *a flow graph*. A weight of edge represents the relative cost of the corresponding link. Distances between positions where the objects should be placed are represented by a weighted undirected graph called *a distance graph*. The objective is to find an one-to-one mapping of vertices of the flow graph on vertices of the distance graph so that the sum of the relative costs is minimized. If the objective is to minimize the maximal cost, the problem is called the quadratic bottleneck assignment problem (QBAP).

The applications of the QBAP include, for example, bandwidth minimization and very large-scale integration. We notice that it often makes sense to minimize the largest cost instead of the overall cost in practice, so any application of the QAP can be an application of the QBAP model.

In this paper we provide polynomial algorithms for some special cases of the QAP and of the QBAP [5, 6, 7].

2 Formulations

Let us provide the formulations of the QAP and of the QBAP. Undirected weighted graphs $LW = (N, E)$ and $MW = (V, U)$ without multiple edges are given. The symbol W means that the graphs are weighted. A weight $w(n_i, n_j) > 0$ is assigned to each edge $\{n_i, n_j\} \in E$ and a weight $r(v_i, v_j) > 0$ is assigned to each $\{v_i, v_j\} \in U$. We assume that $|N| = |V| = m$. A placement of the flow graph LW on the distance graph MW is defined as an one-to-one mapping $\pi : N \to V$. The objective function of the QAP is

$$F(\pi) = \sum_{(n_i,n_j)\in E} w(n_i, n_j)\rho(\pi(n_i), \pi(n_j)) \to \min_{\pi},$$

where $\rho(\pi(n_i), \pi(n_j))$ is the shortest distance between vertices $\pi(n_i)$ and $\pi(n_j)$ in the graph MW.

The objective function of the QBAP is

$$F_b(\pi) = \max_{(n_i,n_j)\in E} w(n_i, n_j)\rho(\pi(n_i), \pi(n_j)) \to \min_{\pi}.$$

We denote the problems by triples (LW, MW, F) and (LW, MW, F_b).

As we mentioned above, the problems are NP-complete in common cases. Furthermore, the

problems (L, Ch, F) and (L, Ch, F_b) of placing an arbitrary unweighted graph L on an unweighted chain Ch are NP-complete [3].

3 Algorithms for QAP

We consider the problem (Ch, TW, F), where $Ch = (N, E)$ is an unweighted chain and $TW = (V, U)$ is a weighted tree. Let the vertices from the set N be renumbered by $\{1, \ldots, m\}$. We denote a degree of the vertex $v \in V$ by $d(v)$.

The idea of the algorithm for solving this problem is to visit each vertex of the tree TW at least once in such way that edges of a constructed path have minimal total weight.

Algorithm 1

Step 1. Find a chain $C_0 = (V_0, U_0)$, $V_0 = (v_1, v_2, \ldots, v_k)$, that has maximal length in the tree TW (i.e. $\rho(v_1, v_k) = \max_{p,q \in V} \rho(p, q)$). Each vertex $v_i \in V_0$ is assigned a subtree $T_i = (V_i, U_i)$. The set V_i consists of v_i and of all tree vertices that are connected with v_i by chains not containing edges from the set U_0. The set U_i consists of edges of mentioned chains.

Step 2. Set $\pi^*(1) = v_1$ and $\pi^*(2) = v_2$. If $d(v_2) = 2$, then $\pi^*(3) = v_3$, else $\pi^*(|V_2| + 2) = v_3$. To place the vertices $\{3, 4, \ldots, |V_2|+1\}$, we use the reduced variant of this algorithm for the subtree T_2, in which there is selected a chain that connects subtree's root with an arbitrary leaf. Examine the other vertices of chain C_0.

The mapping π^* obtained by the algorithm is the optimal solution of the problem. The problem (Ch, TW, F) can be solved using the algorithm 1 in $O(m^3)$ operations.

Let $Cy = (N, E)$ be an unweighted cycle. The problem (Cy, TW, F) is similar to the problem (Ch, TW, F). We "cut" the cycle across an arbitrary edge. We place obtained chain using the modified algorithm 1, in which an arbitrary chain connecting two leaves is selected at step 1. So, the problem (Cy, TW, F) can be solved in $O(m)$ operations.

Definition 1 *A graph $S = (V, U)$ is called a star if there is only one vertex $v_1 \in V$ that has a degree $d(v_1) = m - 1$ and $d(v) = 1$ holds for all vertices $v \in V \setminus \{v_1\}$. The vertex v_1 is called the center of the star.*

Let $LW = (N, E)$ be an arbitrary weighted graph and $SW = (V, U)$ be a weighted star. We provide the algorithm to build the optimal solution of the problem (LW, SW, F).

Algorithm 2

Step 1. Let $Z(n_i) = \sum_{j=1,2,\ldots,m} w(n_i, n_j)$, $i = 1, 2, \ldots, m$. Arrange vertices of the graph LW in such way that $Z(n_1) \geq Z(n_2) \geq \ldots \geq Z(n_m)$ holds.

Step 2. Arrange vertices of the star SW in such way that $r(v_1, v_2) \leq r(v_1, v_3) \leq \ldots \leq r(v_1, v_m)$ holds, where v_1 is the center of the star.

Step 3. Set $\pi^*(n_1) = v_1$, $\pi^*(n_2) = v_2$, $\ldots$, $\pi^*(n_m) = v_m$.

The problem (LW, SW, F) can be solved using the algorithm 2 in $O(m^2)$ operations.

Consider the problem (SW, MW, F), where $MW = (V, U)$ is an arbitrary weighted graph. Using the following algorithm, one can find the optimal solution of the formulated problem.

Algorithm 3

Step 1. Examine all pairs of vertices of the graph MW and build a matrix of shortest paths between them. Zero diagonal elements are neglected in the matrix.

Step 2. Arrange elements of rows of the matrix in nondescending order. Denote resulted matrix by $K = (k_{ij})$, $i = 1, \ldots, m$, $j = 1, \ldots, m - 1$.

Step 3. Arrange vertices of the star SW in such way that $w(n_1, n_2) \geq w(n_1, n_3) \geq \ldots \geq w(n_1, n_m)$ holds, where n_1 is the center of the star.

Step 4. Examine each row of the matrix K, find component-wise products of its elements and vector of the star edges weights. Calculate sums of obtained products. Find the minimum among all these sums. Let the minimum be attained at the row corresponding to the vertex v_1 and the columns in this row correspond to the vertices $v_2, v_3, \ldots, v_m$.

Step 5. Set $\pi^*(n_1) = v_1$, $\pi^*(n_2) = v_2$, $\ldots$, $\pi^*(n_m) = v_m$.

The problem (SW, MW, F) can be solved using the algorithm 3 in $O(m^3)$ operations.

Theorem 1 *The problems* (Ch, TW, F), (Cy, TW, F), (LW, SW, F) *and* (SW, MW, F) *can be solved in polynomial time.*

4 Algorithms for QBAP

We consider the QBAP of placing an unweighted chain $Ch = (N, E)$ on an unweighted tree $T = (V, U)$. In our notation, it is the problem (Ch, T, F_b). To describe the idea of the algorithm for this problem, we need to propose few definitions.

Let the chain $C_0 = (V_0, U_0)$ be selected which consists of the vertices $\{k_1, \ldots, k_p\}$ in the tree T. We call this chain as the *main* one and the tree with selected main chain as *representation* of the tree T with the main chain C_0. The subtrees $T_i = (V_i, U_i)$ are defined in same way as in the problem (Ch, TW, F).

Definition 2 *Let there be a number* $i = 2, \ldots, p-1$ *and there be vertices* $v_r, v_{r+1}, v_s, v_t \in V_i$ *that* v_{r+1} *is adjacent to* v_r, *inequalities* $\rho(v_{r+1}, k_i) > \rho(v_r, k_i)$, $\rho(v_s, v_{r+1}) < \rho(v_s, v_r)$, $\rho(v_t, v_{r+1}) < \rho(v_t, v_r)$ *and* $\rho(v_{r+1}, v_s) = \rho(v_{r+1}, v_t) = 2$ *hold and chains connecting* v_{r+1} *and* v_s, v_{r+1} *and* v_t *do not intersect. A graph consisting of vertices and of edges of chains connecting* v_r *and* v_s, v_r *and* v_t *is called a* G_1*-graph.*

Definition 3 *Let there be a number* $i = 2, \ldots, p-1$ *and there be vertices* $v_r, v_s, v_t \in V_i$ *that* $\rho(k_i, v_r) = \rho(k_i, v_s) = \rho(k_t, v_t) = 2$ *holds and chains connecting these vertices and* k_i *do not intersect. A graph consisting of vertices and of edges of chains connecting* k_i *and* v_r, k_i *and* v_s, k_i *and* v_t *is called a* G_2*-graph.*

Algorithm 4

Step 1. Examine all representations of the tree T with different main chains connecting leaves of the tree. If there are not found both G_1-graph and G_2-graph in some representation of T, go to step 2 with subject to this representation of the tree. Else, go to step 2 with subject to any representation.

Step 2. Apply the special algorithm A_P to place vertices of the chain Ch in vertices of the tree T.

The algorithm A_P is bulky so we describe only its idea which is to place the vertices of Ch in the vertices of the main chain C_0 one by one until $d(k_i) \geq 3$ holds for the current vertex $k_i \in V_0$. To place the vertices in the subtree T_i, one should examine the vertices of T_i moving from the root to the leaves and backward and place the vertices keeping the distance between adjacent ones equals to 2. Then examine the next vertices of the main chain until $d(k_i) \geq 3$ holds again.

The problem (Ch, T, F_b) can be solved using the algorithm 4 in $O(m^4 \log m)$ operations.

Let us consider the problem (SW, MW, F_b), where SW is a weighted star and MW is a weighted graph. We notice that the algorithm 3 can be used for solving this problem if we modify it in such way that at step 4 not the sums of the component-wise products are calculated but the maximum of them. Therefore, the problem (SW, MW, F_b) can be solved in $O(m^3)$ operations.

We describe the algorithm to build the optimal solution π^* of the problem (Ch, SW, F_b).

Algorithm 5

Step 1. Arrange vertices of the star SW in such way that $r(v_1, v_2) \geq r(v_1, v_3) \geq \ldots \geq r(v_1, v_m)$, where v_1 is the center of the star.

Step 2. Set $\pi^*(n_1) = v_2$, $\pi^*(n_2) = v_1$, $\pi^*(n_3) = v_3$, $\pi^*(n_4) = v_n$, $\pi^*(n_5) = v_4$ and so on. The formulas for n_i, $i \geq 4$, are $\pi^*(n_{2j}) = v_{m-j+2}$, $\pi^*(n_{2j+1}) = v_{j+2}$, $j = 2, 3, \ldots, [\frac{m}{2}]$.

The problem (Ch, SW, F_b) can be solved using the algorithm 5 in $O(m \log m)$ operations.

Definition 4 *A tree* $Sp = (V, U)$ *is called a spider if there is only one vertex which degree is greater than* 2 *(spider's center). Chains connecting the center with leaves are called spider legs.*

Let $Sp = (V, U)$ be an unweighted spider. We provide the algorithm for the problem (Ch, Sp, F_b).

Algorithm 6

Step 1. Select an arbitrary leg of the spider Sp with number of vertices not less than 2. Place

vertices of the chain Ch one by one in the vertices of this leg moving from the leaf to the center (the center remains free, i.e. non-occupied).

Step 2. If there are free vertices $v_i \in V$ adjacent to the spider's center that $d(v_i) = 1$ holds, place the vertices of Ch in them. If there are not and the center is free, place the vertex in arbitrary leg's vertex $v_s \in V$ which is adjacent to the center. If the center is occupied and there is only one free leg, place the vertex in v_s and go to step 4. Else the vertex is placed in $v_t \in V$, where v_t is adjacent to v_s.

Step 3. Place the vertices of the chain in the vertices of the leg keeping the distance between occupied spider's vertices equal to 2 and moving to the leaf. Then, place the chain's vertices in same way moving back to the center of the spider. After the last vertex of the leg has been occupied, place the chain's vertex in the spider's center if it is still free. Go to step 2.

Step 4. Place the vertices of Ch in the leg's vertices one by one moving to the leaf.

The problem (Ch, Sp, F_b) can be solved using the algorithm 6 in $O(m)$ operations.

Let us consider the problem of placing an unweighted cycle Cy on an unweighted spider Sp. We place an arbitrary vertex of Cy in the center of Sp at step 1 of the algorithm 6, we eliminate check if there is only one free leg at step 3 and eliminate step 4. The obtained algorithm ends when all the cycle vertices have been placed in the spider vertices. So, the problem (Cy, Sp, F_b) can be solved in $O(m)$ operations.

Theorem 2 *The problems* (Ch, T, F_b), (SW, MW, F_b), (Ch, SW, F_b), (Ch, Sp, F_b) *and* (Cy, Sp, F_b) *can be solved in polynomial time.*

5 Conclusions

The quadratic assignment problem and the quadratic bottleneck assignment problem in terms of graphs are considered. Polynomial algorithms for special flow graphs and distance graphs are provided. In particular, there are algorithms for cases when the flow graph is an unweighted chain and the distance graph is a weighted tree in the first problem and the distance graph is an unweighted tree in the second one.

Acknowledgement. The research was supported by the Russian Foundation for Basic Research (project No. 10-01-00598).

References

[1] R. E. Burkard, M. Dell'Amico and S. Mortello, *Assignment Problems.* Philadelphia: SIAM, 2009.

[2] R. Z. Farahani and M. Hekmatfar, *Facility Location: Concepts, Models, Algorithms.* Heidelberg: Physica-Verlag, 2009.

[3] M. R. Garey and D. S. Johnnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[4] T. C. Koopmans and M. Beckman, *Assignment Problems and the Location of Economic Activities* // Econometric, 1957. No. 25. P. 53–76.

[5] A. Yu. Lagzdin and G. G. Zabudsky, *Polynomial Algorithms for Some Cases of Quadratic Assignment Problem* // Abstracts of International Conference "Operations Research 2010". Munich, 2010. P. 118–119.

[6] G. G. Zabudskii and A. Yu. Lagzdin, *Polynomial Algorithms for Solving the Quadratic Assignment Problem on Networks* // Computational Mathematics and Mathematical Physics. 2010. Vol. 50. No. 11. P. 1948–1955.

[7] G. G. Zabudsky and A. Yu. Lagzdin, *Effective Algorithms for Solving Special Cases of the Quadratic Assignment Problem on Networks* // Proceedings of 8th International Conference "Intelligent Information Processing". M.: MAKS Press, 2010. P. 255–257 (in Russian).