UDC 004.89:004.4

O(1) DELTA PART COMPUTATION TECHNIQUE FOR THE QUADRATIC ASSIGNMENT PROBLEM

S.V. PODOLSKY, YU.M. ZORIN

The quadratic assignment problem is rightfully considered to be one of the most challenging problems of combinatorial optimization. Since this problem is NP-hard, the use of heuristic algorithms is the only way to find in a reasonable time a solution that is close to optimal. One of the most effective heuristic algorithms is the Robust Tabu Search, which is the basis of many subsequent metaheuristic algorithms. The paper describes a novel approach to scanning the neighborhood of the current solution that allows reducing by half the number of delta values that were required to be computed with complexity $O(N)$ in most of the heuristics for the quadratic assignment problem. Using the correlation between the old and new delta values, obtained in this work, a new formula of complexity $O(1)$ is proposed. The results obtained leads up to 25% performance increase as compared to such well-known algorithms as the Robust Tabu Search and others based on it. The formula obtained in this paper may be successfully applied to other heuristics using a full scan of the solution neighborhood.

INTRODUCTION

The quadratic assignment problem (QAP) was first mentioned by Koopmans and Beckmann in 1957 [1] and still remains one of the most challenging combinatorial optimization problems. The problem formulation is as following. There are N locations and N facilities. Distances between each pair of locations and flow values, i.e. number of transportations, are provided. The goal is to assign all the facilities to different locations so that to minimize the sum of all distances multiplied by the corresponding flows. This can be formulated in a form of the objective function

$$\min \sum_{i=1}^{N} \sum_{j=1}^{N} d_{ij} f_{\pi(i)\pi(j)}, \qquad (1)$$

where $\pi(x)$ is a facility number assigned to location with number x; d_{ij} is a distance between locations i and j; $f_{\pi(i)\pi(j)}$ is a flow between facilities $\pi(i)$ and $\pi(j)$. Since the problem is NP-hard [2] there is no exact algorithm that can solve the QAP in polynomial time. Furthermore, the travelling salesman problem (TSP) may be seen as a special case of the QAP in the case when all the facilities are connected via flows having a constant value into a single ring while the other flows have zero value. Thus the QAP can be considered even more challenging than the TSP.

The only methods that allow obtaining feasible solutions for the QAP instances of size 30 and higher are heuristics. One of the most efficient heuristic algorithm for the QAP is the Robust Tabu Search (Ro-TS) developed by Eric

Taillard in 1991 [3], which produces high quality solutions even for very large instances. Among other successful heuristics are genetic algorithms [4], ant systems [5] and others. Typically, predominant majority of them are based on the same approach of the neighbor solutions representations obtained by pairwise exchange of elements in the solution vector.

The purpose of this study is to propose a technique of the current solution neighborhood scanning that allows obtaining a significant performance increase of the metaheuristic algorithm for the quadratic assignment problem solution.

BACKGROUND OF THE NEIGHBORHOOD SCANNING

The process of the neighborhood scanning described for the Ro-TS is one of the most representative environment subject to improvement, since the Ro-TS performs initial solution construction and objective function (1) evaluation only once at the beginning of the algorithm before main iterations started. During the initial stage, a random solution vector is generated and its cost is computed by the objective function as defined by (1). In each main iteration of the Ro-TS a move which relies on exchange of the two elements from the current solution vector is performed. A pair of elements to be exchanged is selected among all C_N^2 pairs using Δ_{ij} minimization criterion. Actually the selection criterion is more complicated than just delta value minimization, but for now we omit its discussion. After the exchange a new cost can be obtained from the previous cost by adding a Δ_{ij} term, where all C_N^2 values Δ_{ij} are computed on the initial stage of the Ro-TS, each with the time complexity $O(N)$, before main iterations start

$$\Delta_{ij} = (d_{ii} - d_{jj})(f_{\pi(j)\pi(j)} - f_{\pi(i)\pi(i)}) + (d_{ij} - d_{ji})(f_{\pi(j)\pi(i)} - f_{\pi(i)\pi(j)}) +$$

$$+ \sum_{g=1, g\neq i,j}^{N} [(d_{gi} - d_{gj})(f_{\pi(g)\pi(j)} - f_{\pi(g)\pi(i)}) + (d_{ig} - d_{jg})(f_{\pi(j)\pi(g)} - f_{\pi(i)\pi(g)})].$$

It is convenient to store all computed values Δ_{ij} in a jagged matrix and extract them by indices i and j once the solution cost must be modified after the elements i and j exchange performed. However, each time after the exchange of two elements, all C_N^2 values Δ_{ij} become unfeasible and needed to be corrected. It is obvious, that after elements p and q are exchanged, the new Δ_{pq} value is evaluated trivially as Δ_{pq} reflecting the reverse exchange of the same pair. For all those Δ_{ij} which do not involve the two elements p and q that were exchanged just before, i.e. $\{i, j\} \bigcap \{p, q\} = \{\varnothing\}$, new values Δ_{ij}^* can be evaluated exploiting their old values Δ_{ij} with the complexity $O(1)$ as following

$$\Delta_{ij}^* = \Delta_{ij} + (d_{pi} - d_{pj} + d_{qj} - d_{qi})(f_{\pi(p)\pi(j)} - f_{\pi(p)\pi(i)} + f_{\pi(q)\pi(i)} - f_{\pi(q)\pi(j)}) +$$

$$+ (d_{ip} - d_{jp} + d_{jq} - d_{iq})(f_{\pi(j)\pi(p)} - f_{\pi(i)\pi(p)} + f_{\pi(i)\pi(q)} - f_{\pi(j)\pi(q)}),\ g \neq i, j. \quad (2)$$

However, as it was observed by Taillard [3], for those Δ_{ij} which involve one of the indices p or q from the last exchange, i.e. $\{i, j\} \bigcap \{p, q\} = 1$ all new values Δ^*_{ij} must be evaluated anew with complexity $O(N)$ by (2). This technique of complete delta recomputation was widely used since it was introduced by Taillard and proceeded without any improvements in further researches such as Reactive Tabu Search [6]. Therefore it is still used in miscellaneous recent Ro-TS implementations such as efficient heuristic for sparse matrices [7] etc. This paper shows how to bypass this issue and evaluate with the linear time complexity only a half of deltas that involve nodes from the last pair exchange. However, it would be important to prove that this issue is a real performance bottleneck and needed to be resolved.

RELEVANCE AND PERFORMANCE

As it was mentioned before, the total number of all possible exchanges equals $C_N^2 = \frac{N(N-1)}{N}$. Among them there are possible pair exchanges that involve only one of the two just exchanged elements and the total number of them equals $2(N-2)$. The last number is obtained from the considerations that each of the two elements being swapped can be exchanged with any element from the rest of $N-2$ other distinct elements, as it shown in Fig. 1. Consequently, the total number of those Δ_{ij} which can be corrected with the complexity $O(1)$ is equal to the difference of all possible exchanges number and the total number of exchanges being corrected with complexity $O(N)$:

$$\frac{N(N-1)}{2} - 2(N-2) = \frac{(N-4)(N-1)}{2} + 2.$$

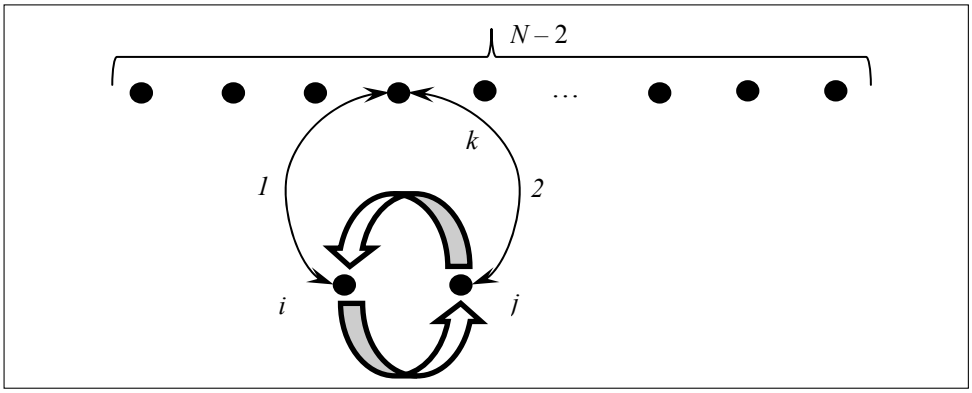
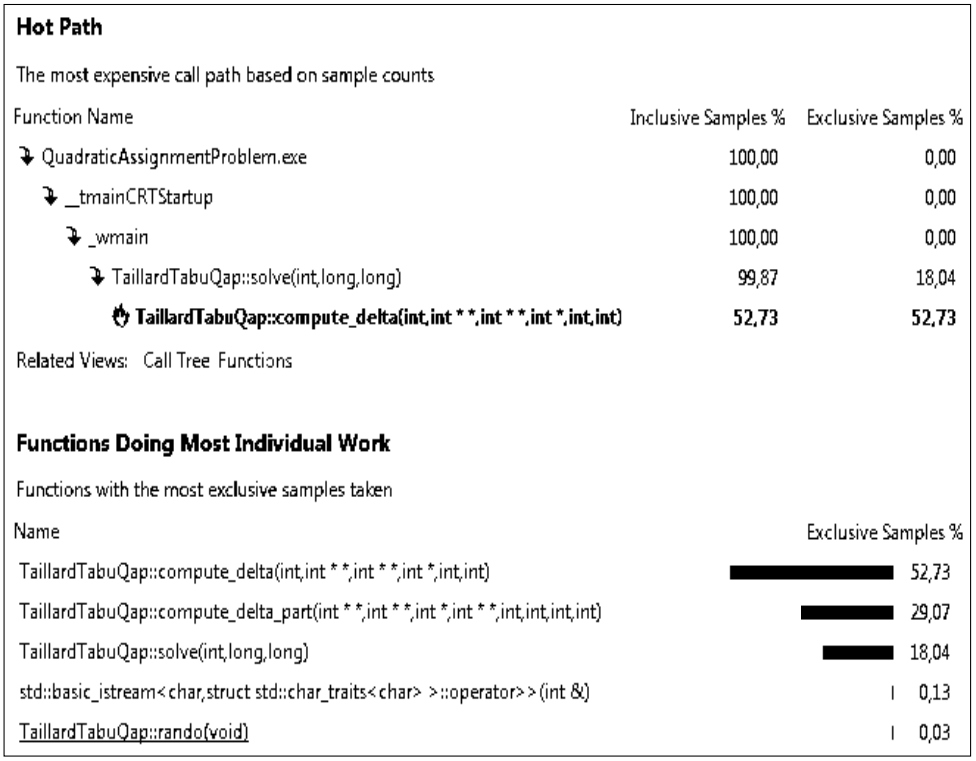


Fig. 1. An illustration of further exchange of one of the two exchanged elements

So, what is to be done is to compute only $(N-2)$ instead of $2(N-2)$ new delta values with complexity $O(N)$ and the rest ones computed with $O(1)$. However, the main question still remains — whether this will affect iteration performance significantly? The answer was obtained after the Ro-TS open source C++ code profiled with the use of MS Visual Studio Profiling tools on `Tai100a` QAP instance.



Hot Path

The most expensive call path based on sample counts

Function Name	Inclusive Samples %	Exclusive Samples %
QuadraticAssignmentProblem.exe	100,00	0,00
_tmainCRTStartup	100,00	0,00
_wmain	100,00	0,00
TaillardTabuQap::solve(int,long,long)	99,87	18,04
TaillardTabuQap::compute_delta(int,int * *,int * *,int *,int,int)	**52,73**	**52,73**

Related Views: Call Tree Functions

Functions Doing Most Individual Work

Functions with the most exclusive samples taken

Name	Exclusive Samples %
TaillardTabuQap::compute_delta(int,int * *,int * *,int *,int,int)	52,73
TaillardTabuQap::compute_delta_part(int * *,int * *,int *,int * *,int,int,int,int)	29,07
TaillardTabuQap::solve(int,long,long)	18,04
std::basic_istream<char,struct std::char_traits<char> >::operator>>(int &)	0,13
TaillardTabuQap::rando(void)	0,03

Fig. 2. A sample profiling report for the Ro-TS on Tai100a instance

As it is shown in Fig. 2, the `compute_delta` method which computes deltas anew takes over 50% of total samples, being the most expensive call, while `compute_delta_part` takes only 29% of total samples to update each of other delta values with the $O(1)$ time complexity. Therefore, we suggest that computational time per iteration can be reduced up to a quarter with the use of proposed technique, especially on large QAP instances.

NEW *O*(1) DELTA PART COMPUTATION TECHNIQUE EXPLORATION

Suppose we have three elements with indices i, j, k in the solution vector and the values $\Delta_{ij}, \Delta_{ik}, \Delta_{jk}$ are already known. Our purpose is to exchange a pair of elements i and j and compute new values $\Delta^*_{ij}, \Delta^*_{ik}, \Delta^*_{jk}$ afterwards. First of all let's consider the assignments* of the flows $f_{\pi(i)\pi(g)}, f_{\pi(j)\pi(g)}, f_{\pi(k)\pi(g)}$ to the distances d_{ig}, d_{jg}, d_{kg} (Fig. 3), where by g we denote an arbitrary element from the rest of the other $N-3$ elements distinct from i, j, k. Though in terms of the QAP an assignment usually means the assignment of facility to specific location, here we imply the assignment of flow to distance caused by a pair of regular QAP assignments. All the values $\Delta_{ij}, \Delta_{ik}, \Delta_{jk}$ include the following terms, which indicate the cost change caused by reassignments of flows to distances to (from) element g after corresponding pair exchanges:

*Though in terms of the QAP an assignment usually means the assignment of facility to specific location, here we imply the assignment of flow to distance caused by a pair of regular QAP assignments.

$$\delta_{ij} = -d_{ig} f_{\pi(i)\pi(g)} - d_{jg} f_{\pi(j)\pi(g)} + d_{ig} f_{\pi(j)\pi(g)} + d_{jg} f_{\pi(i)\pi(g)},$$

$$\delta_{ik} = -d_{ig} f_{\pi(i)\pi(g)} - d_{kg} f_{\pi(k)\pi(g)} + d_{ig} f_{\pi(k)\pi(g)} + d_{kg} f_{\pi(i)\pi(g)},$$

$$\delta_{jk} = -d_{jg} f_{\pi(j)\pi(g)} - d_{kg} f_{\pi(k)\pi(g)} + d_{jg} f_{\pi(k)\pi(g)} + d_{kg} f_{\pi(j)\pi(i)}.$$

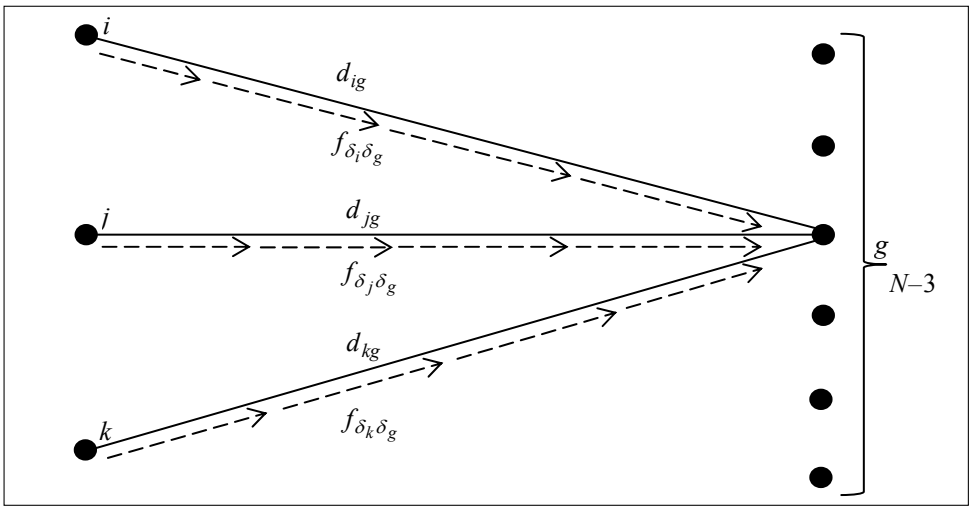


Fig. 3. Assignments of the flows to the distances before any exchanges

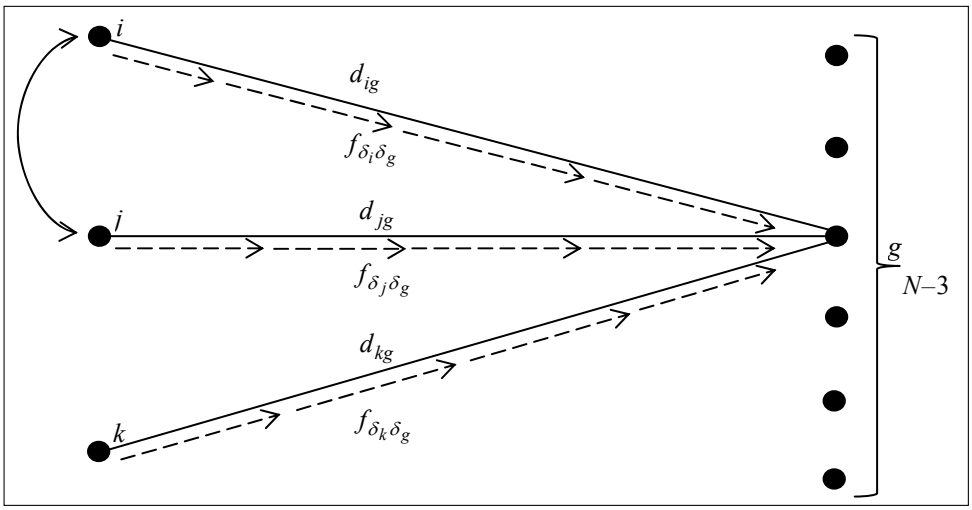


Fig. 4. Assignments of the flows to the distances after *i* and *j* exchanges

Now, let's consider the assignments of the same three flows to the same three distances after the elements i and j are exchanged (see Fig. 4). The new values Δ^*_{ij}, Δ^*_{ik}, Δ^*_{jk} that we need to compute after this exchange will contain the following terms

$$\delta^*_{ij} = -\delta_{ij},$$

$$\delta^*_{ik} = -d_{ig} f_{\pi(j)\pi(g)} - d_{kg} f_{\pi(k)\pi(g)} + d_{ig} f_{\pi(k)\pi(g)} + d_{kg} f_{\pi(j)\pi(g)},$$

$$\delta^*_{jk} = -d_{jg} f_{\pi(i)\pi(g)} - d_{kg} f_{\pi(k)\pi(g)} + d_{jg} f_{\pi(k)\pi(g)} + d_{kg} f_{\pi(i)\pi(g)}.$$

It is important to note, that exactly these $N-3$ terms δ_{ik}^* and δ_{jk}^* for each arbitrary element g require the complexity $O(N)$ for each new Δ_{ik}^* and Δ_{jk}^* values evaluation.

After a simple analysis of the right-hand side expressions for $\delta_{ij}, \delta_{ik}, \delta_{jk}, \delta_{ik}^*, \delta_{jk}^*$ the following dependency between them has been discovered

$$\delta_{ik}^* + \delta_{jk}^* = \delta_{ik} + \delta_{jk} - \delta_{ij}. \tag{3}$$

This means that it is actually not necessary to compute both new values δ_{ik}^* and δ_{jk}^* simultaneously. It is enough to compute anew only one of them and the second one can be obtained with the use of the first.

However, it is worth reminding that we have considered only those terms $\delta_{ij}, \delta_{ik}, \delta_{jk}$ that connect our elements with each of the rest of the other $N-3$ elements. So let's consider now the terms $R_{ij}, R_{ik}, R_{jk}, R_{ik}^*, R_{jk}^*$ which denote the cost change caused by assignments of the flows to distances between the elements i, j, k. The variables R_{ij}, R_{ik}, R_{jk} indicate the cost change before elements i and j are swapped, and the variables R_{ik}^*, R_{jk}^* indicate the cost change after the exchange of i and j respectively

$$\Delta_{ij} = \delta_{ij} + R_{ij},$$

$$\Delta_{ik} = \delta_{ij} + R_{ik},$$

$$\Delta_{jk} = \delta_{jk} + R_{jk},$$

$$\Delta_{ik}^* = \delta_{ik}^* + R_{ik}^*,$$

$$\Delta_{jk}^* = \delta_{jk}^* + R_{jk}^*.$$

The values $R_{ij}, R_{ik}, R_{jk}, R_{ik}^*, R_{jk}^*$ are expressed by the following formulae:

$$R_{ij} = (d_{ik} - d_{jk})(f_{\pi(i)\pi(k)} - f_{\pi(j)\pi(k)}) + (d_{ki} - d_{kj})(f_{\pi(k)\pi(i)} - f_{\pi(k)\pi(j)}) +$$

$$+ (d_{ii} - d_{jj})(f_{\pi(i)\pi(i)} - f_{\pi(j)\pi(j)}) + (d_{ij} - d_{ji})(f_{\pi(i)\pi(j)} - f_{\pi(j)\pi(i)}); \tag{4}$$

$$R_{ik} = (d_{ij} - d_{kj})(f_{\pi(k)\pi(i)} - f_{\pi(j)\pi(i)}) + (d_{ji} - d_{jk})(f_{\pi(i)\pi(k)} - f_{\pi(i)\pi(j)}) +$$

$$+ (d_{ii} - d_{kk})(f_{\pi(k)\pi(k)} - f_{\pi(j)\pi(j)}) + (d_{ki} - d_{ik})(f_{\pi(j)\pi(k)} - f_{\pi(k)\pi(j)})|; \tag{5}$$

$$R_{jk} = (d_{ki} - d_{ji})(f_{\pi(i)\pi(j)} - f_{\pi(k)\pi(j)}) + (d_{ik} - d_{ij})(f_{\pi(j)\pi(i)} - f_{\pi(j)\pi(k)}) +$$

$$+ (d_{jj} - d_{kk})(f_{\pi(k)\pi(k)} - f_{\pi(i)\pi(i)}) + (d_{kj} - d_{jk})(f_{\pi(i)\pi(k)} - f_{\pi(k)\pi(i)}); \tag{6}$$

$$R_{ik}^* = (d_{ij} - d_{kj})(f_{\pi(k)\pi(j)} - f_{\pi(i)\pi(j)}) + (d_{ji} - d_{jk})(f_{\pi(j)\pi(k)} - f_{\pi(j)\pi(i)}) +$$

$$+(d_{ii}-d_{kk})(f_{\pi(k)\pi(k)}-f_{\pi(i)\pi(i)})+(d_{ik}-d_{ki})(f_{\pi(k)\pi(i)}-f_{\pi(i)\pi(k)}); \quad (7)$$

$$R^*_{jk}=(d_{ij}-d_{ki})(f_{\pi(k)\pi(i)}-f_{\pi(j)\pi(i)})+(d_{ij}-d_{ik})(f_{\pi(i)\pi(k)}-f_{\pi(i)\pi(j)})+$$

$$+(d_{jj}-d_{kk})(f_{\pi(k)\pi(k)}-f_{\pi(j)\pi(j)})+(d_{jk}-d_{kj})(f_{\pi(k)\pi(j)}-f_{\pi(j)\pi(k)}). \quad (8)$$

Hence, we can represent the equality (3) as following

$$(\Delta^*_{ik}-R^*_{ik})+(\Delta^*_{jk}-R^*_{jk})=(\Delta_{ik}-R_{ik})+(\Delta_{jk}-R_{jk})-(\Delta_{ij}-R_{ij}).$$

Now we can compute a new value Δ^*_{jk} with the complexity $O(1)$ using the computed anew value Δ^*_{ik} and the old values $\Delta_{ij}, \Delta_{ik}, \Delta_{jk}$

$$\Delta^*_{jk}=\Delta_{ik}+\Delta_{jk}-\Delta_{ij}-\Delta^*_{ik}-R_{ik}-R_{jk}+R_{ij}+R^*_{ik}+R^*_{jk}. \quad (9)$$

After substitution of the expressions (4-8) for $R_{ij}, R_{ik}, R_{jk}, R^*_{ik}, R^*_{jk}$ into (10) which are the parts of (9) without deltas, we will obtain a cumbersome expression (Appendix A) which probably has no benefits on small QAP instances because of a bunch of arithmetical operations

$$-R_{ik}-R_{jk}+R_{ij}+R^*_{ik}+R^*_{jk}. \quad (10)$$

To get rid of such complicated representation, an attempt of simplification has been made with the MATLAB script using `simplify` embedded function (Appendix B). As a result the final compact formula has been obtained to compute all the new values Δ^*_{jk} with the time complexity $O(1)$.

$$\Delta^*_{jk}=\Delta_{jk}+\Delta_{ik}-\Delta_{ij}-\Delta^*_{ik}-$$

$$-(d_{ij}-d_{ik}-d_{ji}+d_{jk}+d_{ki}-d_{kj})(f_{ij}-f_{ik}-f_{ji}+f_{jk}+f_{ki}-f_{kj}).$$

EXPERIMENTAL RESULTS

In order to verify the proposed approach the sample problems provided by E.Taillard and R.Kothari [8] have been chosen. The comparison of the following Taillard's algorithms: the Ro-TS, the Fast ant system (FANT), Tabu Search with Local Optimization (TsLo) with the proposed algorithm (PzTabu) has been done. These algorithms have been chosen because of their high repute and availability of their source codes. All algorithms have been run 50 times on problems with dimensions varying from 20 to 81. The goal of the first part of the simulation was to define best and average solution costs obtained by all algorithms. The second part of simulation has been aimed on comparison of the average time and iterations number required by the algorithms to reach the result not worse than the best one obtained on the first stage. It may be considered as some measure of the algorithm convergence speed. All collected data is shown in Table.

Table. Statistical results of comparative analysis of the algorithms for the QAP solution

Problem	Algo-rithm	Best cost	Mean cost	Mean time to reach the best cost (ms)	Mean number of iterations to reach the best cost
Tai20a	Ro-TS	703482	704449	1353	60175
	FANT	708654	709059	3394	116938
	TsLo	703482	710428	9481	17345
	PzTabu	703482	704296	870	39938
Tai30a	Ro-TS	1818442	1823408	6035	83488
	FANT	1841298	1894374	68786	107317
	TsLo	1825384	1825626	28016	16987
	PzTabu	1820934	1821489	3055	48741
Tai60a	Ro-TS	7265144	7269162	269441	1353407
	FANT	7350644	7351273	553179	1759429
	TsLo	7388361	7396978	777978	65547
	PzTabu	7270382	7274858	159796	879714
Sko81	Ro-TS	91062	91086	241003	560526
	FANT	91106	91323	289203	720510
	TsLo	91596	91666	323767	15944
	PzTabu	91030	91061	135472	360150

From Table we can make sure that the proposed approach outperformed all compared algorithms in terms of operation speed and convergence. It should be noted that, as expected, the larger the size of the problem the greater is the advantage of the proposed approach. A relatively small number of iteration of the TsLo algorithm may be explained by the fact that its each iteration performs local optimization procedure which takes longer time.

CONCLUSIONS

The formula obtained in this paper can also be successfully applied to the other heuristics that use the whole neighbor solutions scanning such as Reactive Tabu Search [6], heuristic for sparse matrices [7] and others. The Ro-TS algorithm is the most representative and exploitable because the solution construction is performed only once and the rest of computational time is dedicated for neighbor solutions scanning and delta values update. As computational experiments proved, we can get significant performance increase in comparison with well-known algorithms including the Ro-TS by replacing a half of $O(N)$ computation operations with the operations of $O(1)$ time complexity.

APPENDIX A. The final formula's part before simplification

$$-R_{ik}-R_{jk}+R_{ij}+R^*_{ik}+R^*_{jk}=$$

$$=(d_{ij}-d_{kj})(f_{\pi(k)\pi(i)}-f_{\pi(j)\pi(i)})+(d_{ji}-d_{jk})(f_{\pi(i)\pi(k)}-f_{\pi(i)\pi(j)})+$$

$$+(d_{ii}-d_{kk})(f_{\pi(k)\pi(k)}-f_{\pi(j)\pi(j)})+(d_{ki}-d_{ik})(f_{\pi(j)\pi(k)}-f_{\pi(k)\pi(j)})-$$

$$
\begin{aligned}
&-(d_{ki}-d_{ji})(f_{\pi(i)\pi(j)}-f_{\pi(k)\pi(j)})+(d_{ik}-d_{ij})(f_{\pi(i)\pi(j)}-f_{\pi(j)\pi(k)})+\\
&+(d_{jj}-d_{kk})(f_{\pi(k)\pi(k)}-f_{\pi(i)\pi(i)})+(d_{kj}-d_{jk})(f_{\pi(i)\pi(k)}-f_{\pi(k)\pi(i)})+\\
&+(d_{ik}-d_{jk})(f_{\pi(i)\pi(k)}-f_{\pi(j)\pi(k)})+(d_{ki}-d_{kj})(f_{\pi(k)\pi(i)}-f_{\pi(k)\pi(j)})+\\
&+(d_{ii}-d_{jj})(f_{\pi(i)\pi(i)}-f_{\pi(j)\pi(j)})+(d_{ij}-d_{ji})(f_{\pi(i)\pi(j)}-f_{\pi(j)\pi(i)})+\\
&+(d_{ij}-d_{kj})(f_{\pi(k)\pi(j)}-f_{\pi(i)\pi(j)})+(d_{ji}-d_{jk})(f_{\pi(j)\pi(k)}-f_{\pi(j)\pi(i)})+\\
&+(d_{ii}-d_{kk})(f_{\pi(k)\pi(k)}-f_{\pi(i)\pi(i)})+(d_{ik}-d_{ki})(f_{\pi(k)\pi(i)}-f_{\pi(i)\pi(k)})+\\
&+(d_{ji}-d_{ki})(f_{\pi(k)\pi(i)}-f_{\pi(i)\pi(j)})+(d_{ij}-d_{ik})(f_{\pi(i)\pi(k)}-f_{\pi(i)\pi(j)})+\\
&+(d_{jj}-d_{kk})(f_{\pi(k)\pi(k)}-f_{\pi(j)\pi(j)})+(d_{jk}-d_{kj})(f_{\pi(k)\pi(j)}-f_{\pi(j)\pi(k)}).
\end{aligned}
$$

APPENDIX B. Matlab formula's simplification script

```
clc
clear all
echo off

syms fii fjj fkk fij fji fik fki fjk fkj
syms dii djj dkk dij dji dik dki djk dkj


Rij = ...
    (dik - djk) * (fik - fjk) + ... % Missed g = k in delta ij
    (dki - dkj) * (fki - fkj) + ... % Missed g = k in delta ij
    (dii - djj) * (fii - fjj) + ... % Loopback
    (dij - dji) * (fij - fji);      % Reverse flows direction

Rik = ...
    (dij - dkj) * (fki - fji) + ... % Missed g = j in delta ik
    (dji - djk) * (fik - fij) + ... % Missed g = j in delta ik
    (dii - dkk) * (fkk - fjj) + ... % Loopback
    (dki - dik) * (fjk - fkj);      % Reverse flows direction

Rjk = ...
    (dki - dji) * (fij - fkj) + ... % Missed g = i in delta jk
    (dik - dij) * (fji - fjk) + ... % Missed g = i in delta jk
    (djj - dkk) * (fkk - fii) + ... % Loopback
    (dkj - djk) * (fik - fki);      % Reverse flows direction

R_ik = ...
    (dij - dkj) * (fkj - fij) + ... % Missed g = j in delta*
ik
    (dji - djk) * (fjk - fji) + ... % Missed g = j in delta*
ik
    (dii - dkk) * (fkk - fii) + ... % Loopback
    (dik - dki) * (fki - fik);      % Reverse flows direction
```

```
R_jk = ...
    (dji - dki) * (fki - fji) + ... % Missed g = i in delta*
jk
    (dij - dik) * (fik - fij) + ... % Missed g = i in delta*
jk
    (djj - dkk) * (fkk - fjj) + ... % Loopback
    (djk - dkj) * (fkj - fjk);      % Reverse flows direction


x = - Rik - Rjk + Rij + R_ik + R_jk;

simplify(x)
```

REFERENCES

1. *Koopmans T.C., Beckmann M.J.* Assignment problems and the location of economic activities // Econometrica. — 1957. — № 25. — P. 53–76.
2. *Sahni S., Gonzalez T.* P-complete approximation problems // Journal of the Association of Computing Machinery. — 1976. — № 23. — P. 555–565.
3. *Taillard E.* Robust tabu search for the quadratic assignment problem // *Parallel Computing*. — 1991. — № 17. — P. 443–455.
4. *Tate D.M.*, *Smith A.E.* A genetic approach to the quadratic assignment problem // *Computers and Operations Research*. — 1995. — № 22. — P. 73–83.
5. *Colorni A., Maniezzo V.* The ant system applied to the quadratic assignment problem // IEEE Transactions on Knowledge and Data Engineering. — 1999. — **11**. — P. 769–778.
6. *Battiti R., Tecchiolli G.* The reactive tabu search // ORSA Journal on Computing. — 1994. — № 6. — P. 126–140.
7. *Paul G.* An efficient implementation of the robust tabu search heuristic for sparse quadratic assignment problems // European Journal of Operational Research. — 2011. — № 209. — P. 215–218.
8. *Kothari R., Ghosh D.* Insertion based Lin-Kernighan heuristic for single row facility layout // Computers & Operations Research. — 2013. — № 40(1). — P. 129–136.

Received 17.07.2013

From the Editorial Board: the article corresponds completely to submitted manuscript.