Journal of Industrial Engineering International
a SpringerOpen Journal

ORIGINAL RESEARCH Open Access

Effective heuristics and meta-heuristics for the quadratic assignment problem with tuned parameters and analytical comparisons

Mahdi Bashiri* and Hossein Karimi

Abstract

Quadratic assignment problem (QAP) is a well-known problem in the facility location and layout. It belongs to the NP-complete class. There are many heuristic and meta-heuristic methods, which are presented for QAP in the literature. In this paper, we applied 2-opt, greedy 2-opt, 3-opt, greedy 3-opt, and VNZ as heuristic methods and tabu search (TS), simulated annealing, and particle swarm optimization as meta-heuristic methods for the QAP. This research is dedicated to compare the relative percentage deviation of these solution qualities from the best known solution which is introduced in QAPLIB. Furthermore, a tuning method is applied for meta-heuristic parameters. Results indicate that TS is the best in 31% of QAPs, and the IFLS method, which is in the literature, is the best in 58 % of QAPs; these two methods are the same in 11% of test problems. Also, TS has a better computational time among heuristic and meta-heuristic methods.

Keywords: Quadratic assignment problem, Heuristics, Meta-heuristics, Tuning method

Background

Quadratic assignment problem (QAP) is a well-known problem which delineates assigning number of economic facility to the same number of location. The objective function of this problem needs to be minimized. This problem was originally initiated by Koopmans and Beckmann (1957). QAP is used in combinational optimization, participating with traveling salesman problem (TSP) and graph problem. QAP is a kind of location problem and, additionally, a kind of layout problem. A lot of research presented exact solutions for QAP in the literature. Figure 1 shows the number of research, which applies the exact methods for QAP until 2005 (Loiola et al. 2007). These methods include branch and bound, dynamic programming, cutting plan, and branch and cut.

Sahni and Gonzalez (1976) proved that QAP belongs to the NP-complete class. In this class of the problem, computational time for solving problems, which have large instances, is too much and grows exponentially, so heuristic approaches and meta-heuristic methods are recommended for solving this problem. Figure 2 shows the literature about solving QAP with heuristic and meta-heuristic methods. This figure reveals that the meta-heuristic methods have been applied more than the heuristic approaches.

For the first time, a simulated annealing (SA) approach in solving QAP was proposed by Burkard and Rendl (1984). Wilhelm and Ward (1987) then presented the new equilibrium components for solving QAP with SA, and after them, research suggested better SA solution methods. Skorin-Kapov (1990) executed the tabu search (TS) technique for QAP. After that, Taillard (1991) demonstrated robust TS for QAP. TS was also proposed for QAP in later research. Particle swarm optimization (PSO) is a new approach to solve QAP that Kennedy and Eberhart (1997), Shi et al. (2004), and Hongbo et al. presented (Hongbo and Ajith 2007; Hongbo et al. 2007). Figure 3 shows the application of meta-heuristic methods for QAP until 2005 (Loiola et al. 2007).

Figure 3 shows that many research used hybrid algorithm for QAP in recent years. Table 1 shows some of the recent research that applied TS, SA, and PSO for QAP from 2007 to 2009.

This paper is organized as follows: in the next section, QAP formulation is displayed; afterwards, the heuristic

* Correspondence: Bashiri@shahed.ac.ir
Department of Industrial Engineering, Shahed University, Tehran 3319118651, Iran

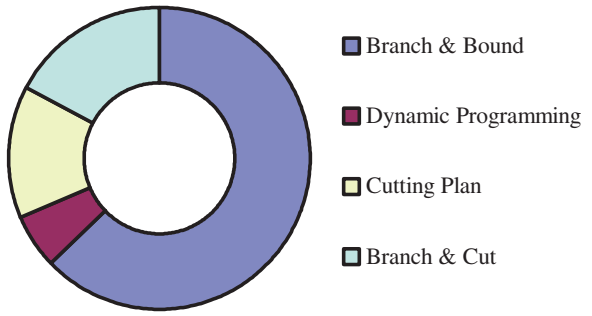

Figure 1 Exact methods for QAP.

methods are described. After that, the meta-heuristic methods such as SA, TS, and PSO are presented; the tuning method is described next. Computational analyses and comparison results are mentioned in the 'Results and discussion' section, and the last section is the 'Conclusions' section.

Methods

QAP formulation

This section is dedicated to introduce the classical formulation of QAP considered in this research. This formulation is as follows:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n} c_{ijkl}x_{ik}x_{jl}, \quad (1)$$

$$\sum_{i=1}^{n} x_{ij}\ , \forall\ j = 1\ldots n, \quad (2)$$

$$\sum_{i=1}^{n} x_{ij}\ , \forall\ i = 1\ldots n, \quad (3)$$

$$x_{ij} = \{0,1\}\ , \forall\ i,j = 1\ldots n, \quad (4)$$

where c_{ijkl} is the cost of assigning facility i in location k and simultaneously facility j in location l, and $x_{ik} = 1$ if location k is assigned to facility i; otherwise, $x_{ik} = 0$. Also, $x_{jl} = 1$ if location l is assigned to facility j; otherwise, $x_{jl} = 0$. The objective function (Equation 1) of this model must be minimized. Each location must be assigned just to one facility, as Equation 2 shows. Equation 3 displays that each facility must be assigned just in one location. The number of facility and location is the same and is equal to n. The variable in this model is binary.

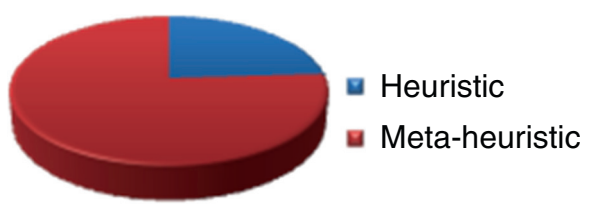

Figure 2 Heuristic and meta-heuristic methods for QAP.

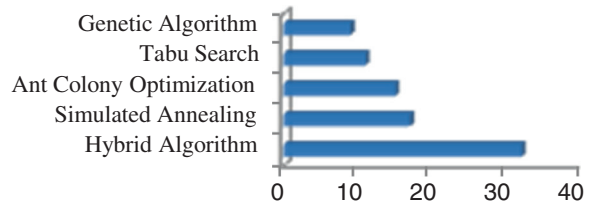

Figure 3 Meta-heuristic approach for QAP.

Heuristic methods

Heuristic algorithms do not provide an assurance for optimization of the problem. These methods are an approximation. They have an additional property that worst-case solutions are known. In this section, some heuristic methods as procedures to search the better solution that contains 2-opt, greedy 2-opt, 3-opt, greedy 3-opt, and Vollman, Nugent, Zartler (VNZ) are contemplated.

2-Opt algorithm

Among simple local search algorithms, the most famous ones are 2-opt and 3-opt. The 2-opt algorithm was first proposed by Croes (1958) for TSP. If there are four locations and four facilities, the transposition of facility location in 2-opt method is like that in Figure 4. This figure illustrates that for the first transposition, the facility in location one can be changed with the facility in location two, and for the second transposition, the facility in location one can be changed with the facility in location three, so that if the number of location and facility is shown by n, the number of transposition in each iteration will be $n(n-1)/2$.

Initially, the algorithm considers the transposition of facilities 1 and 2. If the resulting solution's objective function value (OFV) is smaller than that of the initial solution, then it is stored as a candidate for future

Table 1 TS, SA, and PSO approach for QAP from 2007 to 2009

Authors	Meta-heuristic		
	TS	SA	PSO
Hongbo and Ajith (2007); Hongbo et al. (2007)	-	-	·
Singh and Sharma (2008)	-	·	-
Zhu et al. (2009a,b); James et al. (James et al. 2007, James et al. 2009); Fescioglu and Kokar (2008)	·	-	-
This research	·	·	·

Dot (·), the authors used the method; hyphen (–), the authors did not use the method.

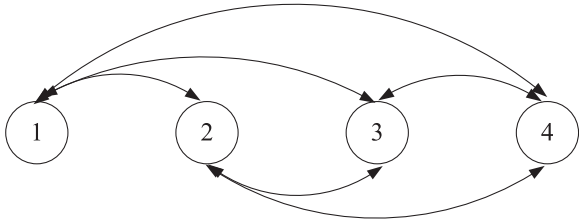

Figure 4 Transposition of facility in the 2-opt method.

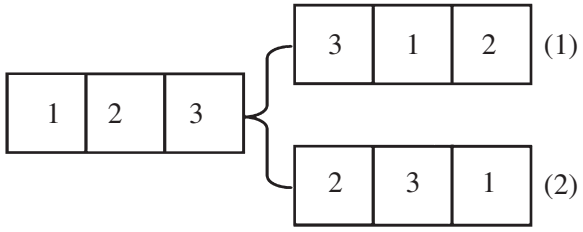

Figure 5 Types of transposition in 3-opt algorithm.

consideration. Otherwise, it is discarded, and the algorithm considers the transposing of facilities 1 and 3. If this exchange generates a better solution, then it is stored as a candidate for future consideration; otherwise, it is discarded, and so on. Thus, whenever a better solution is found, the algorithm discards the previous best solution. This procedure continues until all the pair-wise exchanges are considered.

For n location in the QAP problem, the 2-opt algorithm consists of three steps:

Step 1. Let S be the initial feasible solution and Z its objective function value; then, set $S^* = S$, $Z^* = Z$, $i = 1$ and $j = i + 1 = 2$.

Step 2. Consider the exchange results in a solution S' that has OFV $Z' < Z^*$, set $Z^* = Z'$ and $S^* = S'$. If $j < n$, then repeat step 2; otherwise, set $i = i + 1$ and $j = i + 1$. If $i < n$, repeat step 2; otherwise, proceed to step 3.

Step 3. If $S \neq S^*$, set $S = S^*$, $Z = Z^*$, $i = 1$, $j = i + 1 = 2$ and go to step 2. Otherwise, output S^* is selected as the best solution, and the process is terminated.

Greedy 2-opt algorithm

The greedy 2-opt algorithm is a variant of the 2-opt algorithm. The difference between this method and 2-opt is in selecting the best transposition. This method transposes the facility location if the OFV is better than the known OFV and stabilizes this assignment; it then goes to transpose the facility location from the start. It also consists of three steps. Like the 2-opt algorithm, greedy 2-opt also considers pair-wise exchanges. Initially, it considers transposing facilities 1 and 2. If the resulting OFV is less than the previous one, two facilities are immediately transposed. Otherwise, the algorithm will go on to facility 3 and evaluate the exchange, and so on until improvement is found. If facilities 1 and 2 are transposed, then the algorithm will take it as an initial solution and will repeat the algorithm until it is impossible to improve the solution any further. Greedy 2-opt algorithm makes the exchange permanent whenever an improvement is found and thus consumes less computational time than the 2-opt algorithm. On the other hand, greedy 2-opt algorithm produces slightly worse solutions than the 2-opt algorithm.

3-Opt algorithm

The 3-opt algorithm is similar to the 2-opt algorithm except that it considers transposing two facilities at a time. This algorithm is originally applied for TSP by Bock (1958). For example, if there are three facilities in the same location, two types of transposition can be used with the 3-opt algorithm. These types are shown in Figure 5. Type (2) is applied in this research.

Greedy 3-opt algorithm

Greedy 3-opt algorithm is also similar to the greedy 2-opt algorithm, but it makes the three facility exchange permanent whenever its resulting OFV is better than the current OFV and repeats the algorithm with the new transposition as the initial solution. The transposition in this method is similar to that in 3-opt.

VNZ method

The VNZ method was introduced by Vollman et al. (1968). This method is using less storage space than 2-opt.

There is not any randomization, and also, these methods cannot orient the current solution to the optimum solution in a limited time. However, the meta-heuristic methods contain a good search approach with a reasonable time. These methods are considered in the next subsection.

Meta-heuristic methods

In the original definition, meta-heuristics are solution methods that manage an interaction between the local improvement procedures and higher level strategies to create a process capable of escaping from local optimum solution and performing a good search of solution space. These methods have also come to include any procedures that employ strategies for overcoming the trap of local optimality in complex solution spaces, especially those procedures that take advantage of one or more neighborhood structures as a means of defining acceptable moves to transformation from one solution to another. In this research, TS, SA, and PSO are applied for the QAP, and their comparison has been done for the selected data sets.

Tabu search

Classical methods often face great difficulty when confronted with hard optimization problems present in real situations. Tabu search (TS), for the first time, was proposed by Glover (1989, 1990). This meta-heuristic approach is, in a theatrical manner, changing the ability of solving problems of practical significance. The pseudo code of TS, which is applied in this research, is as follows:

Step 1. Let S be the initial feasible solution and Z its objective function value; then, set $S^* = S$, $Z^* = Z$, max short-term memory (STM) = 5, and max iteration = 1,000; iter = 1. Best O value = O value.
Step 2. Random (i, j) = rand/Long-term memory (LTM) (i, j), ($n1$, $n2$) = the indices of maximum value in random.
Step 3. If there is none ($n1$, $n2$) in STM matrix, change $n1$ and $n2$ locations; otherwise, repeat step 2.
Step 4. Insert $n1$ and $n2$ in STM and release the last indices from STM (e.g., $m1$, $m2$); and LTM($m1$, $m2$) = LTM($m1$, $m2$) + 1.
Step 5. Calculate the objective function value (Z) of the new permutation.
Step 6. If $Z \le Z^*$, then $Z^* = Z$, $S^* = S$, and iter = iter + 1.
Step 7. If iter ≤ max iteration, then repeat step 2; otherwise, print Z^* and S^*.

Simulated annealing

Simulated annealing is a famous and popular local search meta-heuristic applied to address discrete and continuous optimization problems. This method, like the other meta-heuristic methods, can be escaping from the local solution. Simulated annealing is so named because of its similarity to the process of physical annealing with solids, in which a crystalline solid is heated and then allowed to cool very slowly until it achieves its most regular possible crystal lattice configuration and thus is free of crystal defects. If the cooling schedule is sufficiently slow, the final configuration results in a solid with such superior structural integrity. Equation 5 shows the Metropolis acceptance criteria for each move in the cooling process (Metropolis et al. 1953).

$$P = \exp\left(\frac{(\mathrm{OFV}_B - \mathrm{OFV})}{T}\right), \tag{5}$$

where OFV and OFV_B are the objective function values for this iteration and are the best computed one until this iteration. T is the temperature of the algorithm in the iteration, and P is the probability of acceptance for each move in the annealing process. The proposed SA pseudo code for QAP is as follows:

Step 1. Let S be the initial feasible solution and Z its objective function value; then, set $S^* = S$, $Z^* = Z$. T = 100, T_0 = 0.1, r = 0.95, $n$limit max = 5 and $n$over max = 10.
Step 2. $n$limit = 0 and $n$over = 0.
Step 3. Transpose two facilities in the current layout randomly and calculate the objective function value (Z).
Step 4. If $Z \le Z^*$, then accept the transposition; it means that $Z^* = Z$, $S^* = S$, and then $n$limit = $n$limit + 1, $n$over = $n$over + 1; if $n$over = $n$over max or $n$limit =

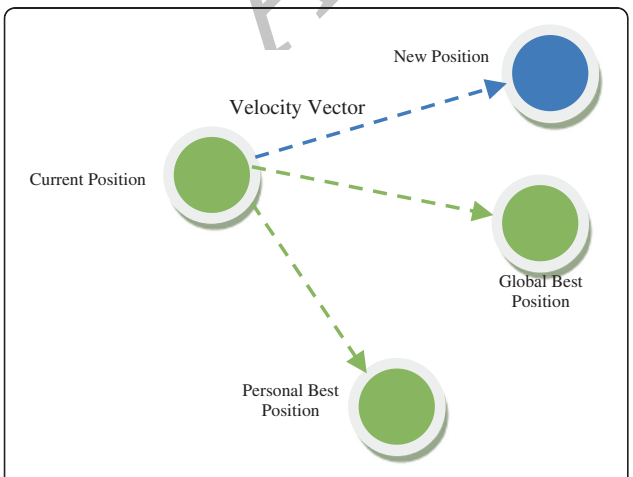

Figure 6 Each particle movement.

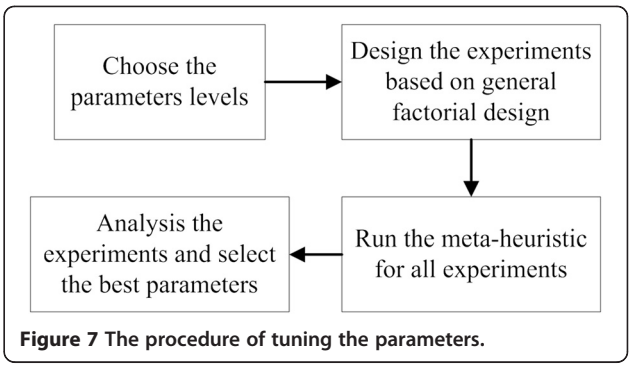

Figure 7 The procedure of tuning the parameters.

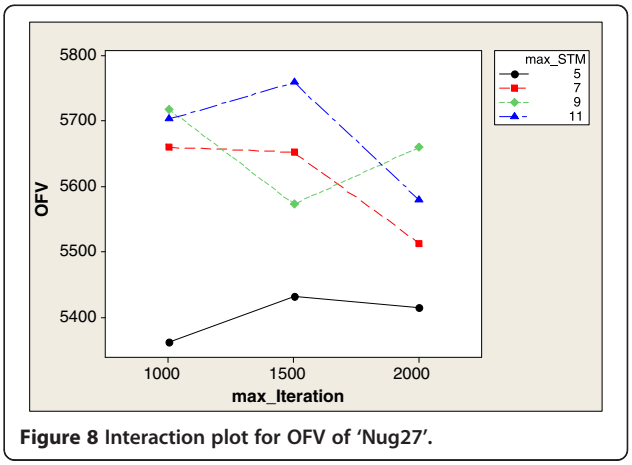

Figure 8 Interaction plot for OFV of 'Nug27'.

*n*limit max, then proceed to step 6.Otherwise, repeat step3; if $Z > Z^*$, then proceed to step 5.

Step 5. Calculate Equation 3; if $P \geq$ rand (0, 1), then $Z^* = Z$, $S^* = S$, and then *n*limit = *n*limit + 1, *n*over = *n*over + 1; if *n*over = *n*over max or *n*limit = *n*limit max, then proceed to step 6; otherwise, repeat step 3.If $P <$ rand (0, 1), then *n*over = *n*over + 1; if *n*over = *n*over max, then proceed to step 6; otherwise, repeat step 3.

Step 6. $T = r \times T$, where r is the rate of cooling. If $T \leq T_0$, then proceed to step 7; otherwise, repeat step 2.

Step 7. Print S^* and Z^*.

Partial swarm optimization

The particle swarm optimization (PSO) is a population-based search algorithm founded on the simulation of the social behavior of bees, birds, or a school of fish. This method is a significant member of the swarm intelligence. It was proposed by Eberhart and Kennedy as an optimization method (Eberhart and Kennedy 1995; Kennedy and Eberhart 1995). Each individual within the swarm is represented by a vector in multidimensional search space. This vector has one assigned vector that determines the next movement of the particle and is called the velocity vector. This technique also determines how to update the velocity of a particle. Each particle updates its velocity based on the current velocity and the best position (*p*best) it has explored so far, as well as based on the global best position (*g*best) explored by a swarm. Movement of each particle is shown in Figure 6, and it is based on Equations 6 and 7. Equation 6 illustrates that the velocity vector is updated by the global best position, personal best position, and current position of each particle. Equation 7 shows that each particle moves by its velocity.

$$v_i(t+1) = w.v_i(t) + b_1.\text{rand}(p\text{best} - x_i(t)) + b_2.\text{rand}(g\text{best} - x_i(t)), \quad (6)$$

$$x_i(t+1) = x_i(t) + v_i(t), \quad (7)$$

where i is the index of the particle, t is the index of an iteration, v_i is the vector of velocity, x_i is the position, w is the weight of current velocity, b_1 is the weight of difference between personal best and current positions, b_2 is the weight of difference between global best and current positions, and rand is used for randomization. The PSO

Table 2 Heuristic computational results for the test problems

Names of instances	*n*	2-Opt		Greedy 2-opt		3-Opt		Greedy 3-opt		VNZ		Exact
		Best OFV	Time (s)	Best OFV	Time (s)	Best OFV	Time (s)	Best OFV	Time (s)	Best OFV	Time (s)	Best OFV
Nug12	12	630	0.035	720	0.070	612	0.060	662	0.038	630	0.115	578
Nug14	14	1040	0.033	1270	0.031	1076	0.120	1122	0.037	1094	0.079	1014
Nug15	15	1168	0.080	1410	0.031	1194	0.158	1266	0.081	1210	0.094	1150
Nug16a	16	1636	0.036	2032	0.061	1710	0.150	1744	0.092	1708	0.160	1610
Nug16b	16	1312	0.062	1478	0.039	1304	0.175	1342	0.047	1338	0.092	1240
Nug17	17	1764	0.043	2204	0.051	1824	0.183	1900	0.052	1812	0.094	1732
Nug18	18	1988	0.033	2438	0.038	2002	0.325	2102	0.078	2084	0.104	1930
Nug20	20	2676	0.095	3070	0.037	2700	0.300	2780	0.089	2762	0.076	2570
Nug21	21	2560	0.055	3224	0.073	2528	0.482	2612	0.145	2568	0.096	2438
Nug22	22	3836	0.050	4580	0.043	3706	0.702	3850	0.070	4002	0.172	3596
Nug24	24	3670	0.051	4550	0.052	3580	0.889	3776	0.121	3826	0.143	3488
Nug25	25	3816	0.056	4746	0.053	3850	0.999	4014	0.194	3946	0.186	3744
Nug27	27	5582	0.069	6436	0.037	5622	1.580	5932	0.214	5792	0.138	5234
Nug30	30	6180	0.081	7450	0.052	6282	3.476	6670	0.260	6648	0.238	6124
Bur26a	26	5536606	0.104	5686514	0.079	5450887	5.318	5499462	0.390	5642823	0.269	5426670
Bur26b	26	3865109	0.201	4078658	0.082	3825543	6.888	3921950	0.390	3898585	0.340	3817852
Bur26c	26	5559892	0.158	5691003	0.089	5432628	6.066	5488940	0.377	5560520	0.300	5426795
Bur26d	26	3955005	0.160	4032499	0.079	3822905	5.472	3845665	0.408	3999274	0.324	3821225
Bur26e	26	5457341	0.131	5656466	0.082	5387936	6.510	5441312	0.382	5610566	0.351	5386879
Bur26f	26	3882393	0.149	4038741	0.079	3783547	4.828	3818354	0.390	3982126	0.282	3782044
Bur26g	26	10178342	0.148	10668053	0.080	10119845	5.332	10243826	0.389	10405839	0.281	10117172
Bur26h	26	7246369	0.101	7644484	0.085	7100009	5.651	7216120	0.391	7286571	0.281	7098658

Table 3 Solution qualities with CPU time of small-size test problems

Names of instances	*n*	SA		TS		PSO		IFLS with local search (OXPM)		Exact
		RPD	Time (s)	RPD	Time (s)	RPD	Time (s)	RPD	Time (s)	OFV
Nug12	12	**0.00**	0.16	**0.00**	**0.12**	7.61	0.27	1.38	0.70	578
Nug14	14	1.78	0.19	**0.39**	**0.17**	1.38	0.29	1.58	1.28	1,014
Nug15	15	3.30	0.16	0.87	**0.15**	2.26	0.27	**0.17**	1.69	1,150
Nug16a	16	0.75	0.16	1.37	**0.14**	0.75	0.31	**0.00**	2.17	1,610
Nug16b	16	0.65	0.17	**0.00**	**0.14**	**0.00**	0.32	**0.00**	2.14	1,240
Nug17	17	1.04	0.17	**0.69**	**0.15**	3.00	0.32	1.27	2.77	1,732
Nug18	18	2.69	0.18	1.04	**0.14**	3.01	0.34	**0.93**	3.42	1,930
Nug20	20	2.49	0.20	1.56	**0.16**	1.95	0.36	**0.70**	5.28	2,570
Nug21	21	1.72	0.20	**0.25**	**0.17**	2.71	0.40	**0.25**	6.89	2,438
Nug22	22	1.78	0.20	**0.50**	**0.17**	1.33	0.41	1.33	7.77	3,596
Nug24	24	4.59	0.21	**1.15**	**0.18**	3.96	0.45	2.87	10.91	3,488

pseudo code, which is presented for QAP in this research, is as follows:

Step 1. max iteration = 100, number of particle = 15, and $w = n - 1$, $b_1 = n/2$ and $b_2 = (n/2) + 2$, where n is the dimension of the problem. Make 15 permutations as the initial solutions and $Z^* = \min$ (OFV), $S^* = S$, and iter = 1.

Step 2. For $i = 1$ to w, transpose two facility. Do this step for each particle.

Step 3. Calculate objective function value (Z) for each new particle; find the personal best OFV for each particle ($p$best), and find the global best OFV ($g$best).

Step 4. Generate a random discrete number between 0 and b_1, and for $i = 1$ to this random number, simulate each particle to $p$best.

Step 5. Generate a random discrete number between 0 and b_2, and for $i = 1$ to this random number, simulate each particle to $g$best.

Table 4 Solution qualities with CPU time of medium-size test problems

Names of instances	*n*	SA		TS		PSO		IFLS with local search (OXPM)		Exact
		RPD	Time (s)	RPD	Time (s)	RPD	Time (s)	RPD	Time (s)	OFV
Nug25	25	3.69	0.24	1.55	**0.18**	1.92	0.48	**0.21**	12.97	3,744
Bur26a	26	0.10	0.62	**0.09**	**0.53**	0.20	1.32	**0.09**	15.86	5,426,670
Bur26b	26	0.67	0.61	0.19	**0.54**	0.42	1.30	**0.17**	15.56	3,817,852
Bur26c	26	0.06	0.60	0.26	**0.55**	0.30	1.34	**0.00**	15.41	5,426,795
Bur26d	26	0.12	0.60	0.02	**0.54**	0.05	1.30	**0.01**	15.38	3,821,225
Bur26e	26	0.01	0.60	**0.03**	**0.51**	0.05	1.33	0.26	15.16	5,386,879
Bur26f	26	0.26	0.61	0.05	**0.54**	0.33	1.29	**0.00**	15.55	3,782,044
Bur26g	26	0.06	0.59	**0.01**	**0.54**	0.42	1.34	**0.01**	15.28	10,117,172
Bur26h	26	0.03	0.60	0.01	**0.52**	0.35	1.32	**0.00**	14.88	7,098,658
Nug27	27	2.67	0.26	**1.38**	**0.22**	4.43	0.48	2.79	17.42	5,234
Nug30	30	5.94	0.27	2.65	**0.22**	6.01	0.54	**1.32**	22.45	6,124
Tai30a	30	4.70	0.27	4.75	**0.25**	5.90	0.85	**2.46**	18.28	1,818,146
Tai30b	30	3.55	0.26	**1.62**	**0.25**	5.18	0.87	2.33	18.23	637,117,113
Tai40a	40	5.31	0.37	6.12	**0.32**	5.53	1.16	**3.16**	62.66	3,139,370
Tai40b	40	5.00	0.34	**3.07**	**0.29**	3.84	1.07	4.18	60.44	637,250,948
Tai50a	50	5.90	0.47	6.49	**0.39**	7.19	1.36	**2.90**	158.94	4,938,796
Tai50b	50	3.86	0.45	5.15	**0.40**	7.23	1.35	**1.87**	161.27	458,821,517

Table 5 Solution qualities with CPU time of large size test problems

Names of instances	*n*	SA		TS		PSO		IFLS with local search (OXPM)		Exact
		RPD	Time (s)	RPD	Time (s)	RPD	Time (s)	RPD	Time (s)	OFV
Lipa80a	80	0.25	0.89	**0.15**	**0.79**	40.70	2.39	0.72	1734.11	253,195
Lipa80b	80	23.46	0.94	23.55	**0.80**	23.93	2.38	**20.65**	1889.81	7,763,962
Tai80a	80	6.13	0.94	6.66	**0.79**	7.01	2.37	**2.87**	1390.58	13,527,910
Tai80b	80	2.93	0.96	3.91	**0.82**	4.55	2.48	**0.71**	1122.56	841,223,593
Lipa90a	90	2.01	1.12	**0.49**	**0.93**	1.03	3.07	0.67	2964.41	360,630
Lipa90b	90	13.55	1.17	14.59	**0.96**	14.71	3.35	**0.00**	3200.88	12,490,441
Tai100a	100	4.66	1.69	4.11	**1.20**	4.33	4.82	**2.69**	3560.03	21,090,402
Tai100b	100	3.06	2.15	2.02	**1.73**	4.17	5.21	**0.93**	3595.17	1.186E + 09

Step 6. iter = iter + 1; if iter < max iteration, then repeat step 2;otherwise, proceed to step 7.
Step 7. Print Z^* = *g*best S^* = *g*best permutation (S).

Tuning method

The parameters in meta-heuristic methods which are introduced, such as max STM and max iteration, were tuned by carrying out the general factorial design. To achieving this purpose, some levels were defined initially for each parameter. These levels are determined to be focused on the logic of meta-heuristics. For instance, when the number of iteration in PSO is increased, then the time of search is intensified. This is just one characteristic of this meta-heuristic approach. After defining the levels, an experimental design is determined by general factorial design. The meta-heuristic for these experiments is then run, and their results are analyzed. Thus, the best parameter level can be found. Figure 7 illustrates this tuning method.

For example, for 'Nug' instances taken from the Quadratic Assignment Problem Library (QAPLIB), the analysis is done for 'Nug27' for the TS method. The proposed levels for max STM are 5, 7, 9, and 11, and for max iteration, they are 1,000, 1,500, and 2,000. We consider two replicates for each combination of factor levels; hence, for this instance, 24 treatment combinations are arranged. Figure 8 shows the interaction plot for OFV, and it also shows the best combination of these factors. The best max STM is 5, and the best max iteration is 1,000. Hence, we tune the TS parameter based on this experiment. Moreover, the parameters like *n*over max, *n*limit max, T, T_0, and others were set by analyzing the general factorial design similar to TS.

Results and discussion

Computational analyses

In this research, an analytical comparison between heuristics and meta-heuristics, which are described as the Nug and 'Bur' instances selected from QAPLIB, are applied. The computer which carried out the experiments is equipped with a 2 53-GHz processor with programming coded in MATLAB7.8 (MathWorks, Natick, MA, USA). Table 2 shows the computational results for heuristic methods. Results show that 3-opt as a heuristic method is the best solution method for QAP among the presented heuristic methods in this research. In this table, sizes of the problems are determined by *n*.

Tables 3, 4, and 5 reveal the computational results for meta-heuristic methods. In this computational analysis, the test problems are classified in small-, medium-, and large-size cases. These groups are taken from the study of Ramkumar et al. (2008). Our experiment results are compared with iterated fast local search (IFLS) when the order crossover with random pair-wise interchange mutation (OXPM) is used (Ramkumar et al 2008). The criterion considered for evaluating the performance is the relative percentage deviation (RPD) of the solution quality from the best known solution. The number of location is *n*.

The superior methods in CPU time and OFV are illustrated in the boldface. The average of gaps for heuristic methods and meta-heuristic methods introduced in this research with the exact solution are shown in Figure 9. These results are related to the Nug and Bur instances.

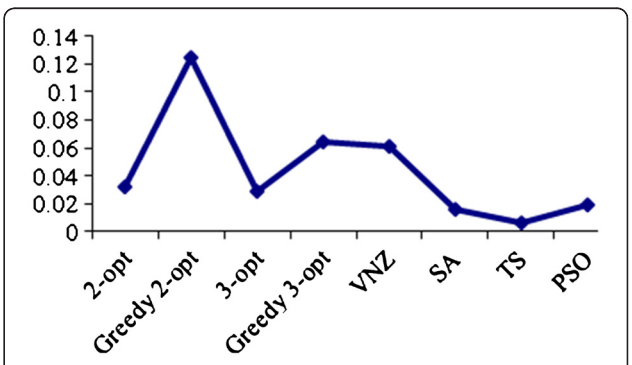

Figure 9 Average gaps for heuristics.

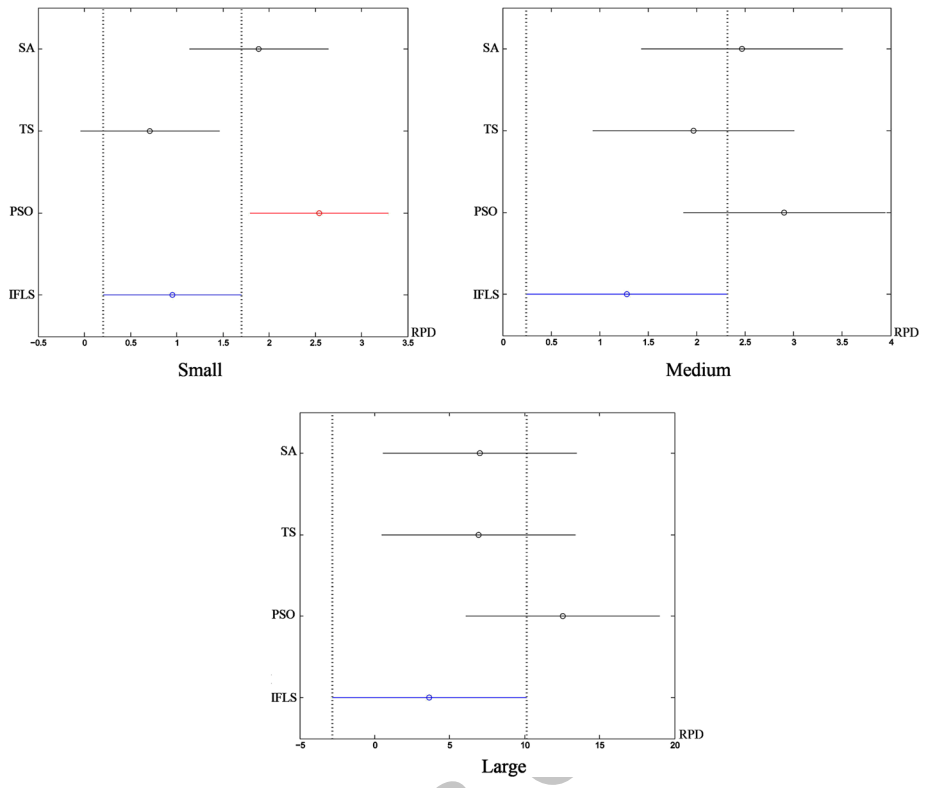

Figure 10 ANOVA results give 95% confidence interval for the difference among meta-heuristics and IFLS.

This figure shows that TS is the best in our methods. In addition, it can be concluded that meta-heuristics have better solution quality; hence, we compare meta-heuristic methods with IFLS. Furthermore, Figure 9 shows that the greedy-2opt is not a good method because the average gap for this method is about 12% in selected data sets.

The performances of our proposed meta-heuristic approaches and IFLS with local search are studied by analysis of variance (ANOVA) test.

Figure 10 summarizes the results of our investigation where the critical value for this analysis is considered as 0.05. As we can observe from Figure 10, in small-size test problems, the means of IFLS and PSO are significantly different, and no meta-heuristics have means significantly different from IFLS in the medium- and large-test problems. It is clear from this figure that TS and IFLS have the best solution quality in small-, medium-, and large-size instances, respectively. Hence, it is not necessary that the p values of these analyses be declared.

Conclusions

This research considers the heuristic and meta-heuristic solution methods for QAP, and the comparison among them has been done. In addition, a tuning method is declared. The comparison has been executed for the selected data set which was extracted from QAPLIB. The results show that 3-opt has better results than the other heuristic methods. Moreover, our meta-heuristic methods are better than the heuristic methods in solution quality. In this paper, some small, medium, and large test problems are used for comparing our meta-heuristic methods to a method from literature (i.e., IFLS with local search). ANOVA test is run for the results, and it showed that our methods are not considerably different from IFLS. TS is the most excellent method in computational time. Comparisons between the selected solution methods for more instances from the QAPLIB and comparisons of these algorithms with other meta-heuristics and hybrid algorithms can be conducted in future research.

Competing interests
The authors declare that they have no competing interests.

Authors' contributions
MB has worked on the literature and modeling; he also proposed the TS, SA, and PSO for his work. HK has performed numerical experiments and carried out sensitivity analysis. Both authors read and approved the final manuscript.

Acknowledgments
The authors would like to thank the anonymous referees for their constructive comments on earlier version of this work.

Received: 22 May 2010 Accepted: 3 March 2012
Published: 18 July 2012

References

Bock F (1958) An algorithm for solving traveling-salesman and related network optimization. Research report associated with talk presented at the Operations Research Society of America 14th National Meeting, St. Louis

Burkard R, Rendl F (1984) A thermodynamically motivated simulation procedure for combinatorial optimization problems. European Journal of Operational Research 17:169–174

Croes GA (1958) A method for solving traveling salesman problems. Operation Research 6:791–812

Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. Paper presented at the Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan

Fescioglu U, Kokar M (2008) Application of Self Controlling Software Approach to Reactive Tabu Search. SASO, Venezia, pp 297–305

Glover F (1989) Tabu search: Part I. ORSA J Comput 1:190–206

Glover F (1990) Tabu search: Part II. ORSA J Comput 2:4–32

Hongbo L, Ajith A (2007) An hybrid fuzzy variable neighborhood particle swarm optimization algorithm for solving quadratic assignment problems. Jucs 13 (9):1309–1331

Hongbo L, Ajith A, Jianying Z (2007) A particle swarm approach to quadratic assignment problems. Soft Comput Ind App 39:213–222

James T, Rego C, Glover F (2007) A cooperative parallel tabu search algorithm for the quadratic assignment problem. Eur J Oper Res 195:810–826

James T, Rego C, Glover F (2009) Multistart tabu search and diversification strategies for the quadratic assignment problem. Systems, Man and Cybernetics, Part A: Systems and Humans 39(3):579–596

Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, Nov-Dec 1995, 4th edn. IEEE, Piscataway, pp 1942–1948

Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. In: Proceedings of International Conference on Systems, Man, and Cybernetics, October 1994, 1st edn. IEEE Computer Society Press, New York, pp 4104–4108

Koopmans T, Beckmann M (1957) Assignment problems and the location of economic activities. Econometrica 25:53–76

Loiola E, de Abreo N, Boaventura-Net P, Hahn P, Querido T (2007) A survey for the quadratic assignment problem. Eur J Oper Res 176:657–690

Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller M (1953) Equation of state calculations for fast computing machines. Journal of Chemical 21: 1087–1092

Ramkumar AS, Ponnambalam SG, Jawahar N, Suresh RK (2008) Iterated fast local search algorithm for solving quadratic assignment problems. Rob Comput-Integr Manuf 24:392–401

Sahni S, Gonzalez T (1976) P-complete approximation problems. J Assoc Comput Mach 23:555–565

Shi X, Xing X, Wang Q (2004) A discrete PSO method for generalized TSP problem. In: Proceedings of International Conference on Machine Learning and Cybernetics, August 2004, 4th edn. IEEE Computer Society Press, New York, pp 2378–2383

Singh SP, Sharma RK (2008) Two-level modified simulated annealing based approach for solving facility layout problem. Int J Prod Res 46(13):3563–3582

Skorin-Kapov J (1990) Tabu search applied to the quadratic assignment problem. ORSA J Comput 2:33–45

Taillard E (1991) Robust taboo search for the quadratic assignment problem. Parallel Comput 17:443–455

Vollman TE, Nugent CE, Zartler RL (1968) A computerized model for office layout. J Ind Eng 19:321–327

Wilhelm MR, Ward TL (1987) Solving quadratic assignment problems by simulated annealing. IEEE Trans 19:107–119

Zhu W, Curry J, Marquez A (2009a) GPU-accelerated simt tabu search for the quadratic assignment problem. NAMRC, Greenville, pp 435–442

Zhu W, Curry J, Marquez A (2009b) SIMD tabu search for the quadratic assignment problem with graphics hardware acceleration. Int J Prod Res 48:1035–1047

doi:10.1186/2251-712X-8-6
Cite this article as: Bashiri and Karimi: **Effective heuristics and meta-heuristics for the quadratic assignment problem with tuned parameters and analytical comparisons.** *Journal of Industrial Engineering International* 2012 **8**:6.