A Particle Swarm Approach to Quadratic Assignment Problems

Hongbo Liu[1,3], Ajith Abraham[1,2], and Jianying Zhang[3]

[1] School of Computer Science, Dalian Maritime University, Dalian, 116024, China
[2] School of Computer Science and Engineering, Yonsei University, 134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749, Korea
ajith.abraham@ieee.org
[3] Department of Computer Science, Dalian University of Technology, Dalian, 116023, China
{lhb,zhangjy}@dlut.edu.cn

Summary. Particle Swarm Optimization (PSO) algorithm has exhibited good performance across a wide range of application problems. But research on the Quadratic Assignment Problem (QAP) has not much been investigated. In this paper, we introduce a novel approach based on PSO for QAPs. The representations of the position and velocity of the particles in the conventional PSO is extended from the real vectors to fuzzy matrices. A new mapping is proposed between the particles in the swarm and the problem space in an efficient way. We evaluate the performance of the proposed approach with Ant Colony Optimization (ACO) algorithm. Empirical results illustrate that the approach can be applied for solving quadratic assignment problems and it has outperforms ACO in the completion time.

1 Introduction

Particle Swarm Optimization (PSO) algorithm is inspired by social behavior patterns of organisms that live and interact within large groups. In particular, PSO incorporates swarming behaviors observed in flocks of birds, schools of fish, or swarms of bees, and even human social behavior, from which the Swarm Intelligence(SI) paradigm has emerged [1, 2]. It could be implemented and applied easily to solve various function optimization problems, or the problems that can be transformed to function optimization problems. As an algorithm, the main strength of PSO is its fast convergence, which compares favorably with many global optimization algorithms [3, 4, 5]. PSO has exhibited good performance across a wide range of applications [6, 7, 8]. However, research on discrete problems, especially Quadratic Assignment Problem (QAP), has been done little [9, 10]. In this paper, we design a fuzzy scheme based on discrete particle swarm optimization [11, 12] to solve quadratic assignment problems.

2 Quadratic Assignment Problem

The quadratic assignment problem (QAP) is a standard problem in location theory. It was introduced by Koopmans and Beckmann in 1957 [13] and is a model for many practical problems [14]. Intuitively, the QAP can be described as the problem of assigning a set of facilities to a set of locations with given distances between the locations and given flows between the facilities. The goal then is to place the facilities on locations in such a way that the sum of the product between flows and distances is minimal. More formally, given n facilities $\{F_1, F_2, \cdots, F_n\}$ and n locations $\{L_1, L_2, \cdots, L_n\}$, two $n \times n$ matrices $FM = [f_{ij}]$ and $DM = [d_{rs}]$, where f_{ij} is the flow between facilities F_i and F_j and d_{rs} is the distance between locations L_r and L_s, the QAP can be stated as follows:

$$\min_{\Pi \in P(n)} Z_{\Pi} = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\Pi_i \Pi_j} \tag{1}$$

where $P(n)$ is the set of all permutations (corresponding to the assignment solutions) of the set of integers $\{1, 2, \cdots, n\}$, and Π_i gives the location of facility F_i in the current solution $\Pi \in P(n)$. Here $f_{ij} d_{\Pi_i \Pi_j}$ describes the cost contribution of simultaneously assigning facility F_i to location Π_i and facility F_j to location Π_j. It is to be noted that the number of facilities (n) is assumed to be the same as the number of locations. In the other word, one facility could be assigned to only one location, and one location could be assigned to only one facility in a feasible assignment solution.

The term *quadratic* stems from the formulation of the QAP as an integer optimization problem with a quadratic objective function [14]. Let b_{ij} be a binary variable which takes value 1 if facility F_i is assigned to location L_j and 0 otherwise. Then the problem can be re-formulated as:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{r=1}^{n} \sum_{s=1}^{n} f_{ij} d_{rs} b_{ir} b_{js} \tag{2}$$

s.t.

$$b_{ij} \in \{0, 1\}, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, n; \tag{3}$$

$$\sum_{i=1}^{n} b_{ij} = 1, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, n; \tag{4}$$

$$\sum_{j=1}^{n} b_{ij} = 1, \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, n. \tag{5}$$

The QAP is a NP-hard optimization problem [15]. While some NP-hard combinatorial optimization problems can be solved exactly for relatively large instances, as exemplified by the traveling salesman problem (TSP), QAP instances of size larger than 20 are considered intractable. The QAP is considered as one of the hardest optimization problems, because exact algorithms show a very poor performance on it [16]. Therefore, several heuristics have been proposed for finding near-optimum solutions for large QAP instances, including ant colonies optimization [17, 18, 19].

3 Particle Swarm Model

The classical PSO model consists of a swarm of particles, which are initialized with a population of random candidate solutions. They move iteratively through the d-dimension problem space to search the new solutions, where the fitness, f, can be calculated as the certain qualities measure. Each particle has a position represented by a position-vector $\mathbf{x}_i$ (i is the index of the particle), and a velocity represented by a velocity-vector $\mathbf{v}_i$. Each particle remembers its own best position so far in a vector $\mathbf{x}_i^{\#}$, and its j-th dimensional value is $x_{ij}^{\#}$. The best position-vector among the swarm so far is then stored in a vector $\mathbf{x}^*$, and its j-th dimensional value is x_j^*. During the iteration time t, the update of the velocity from the previous velocity to the new velocity is determined by Eq.(6). The new position is then determined by the sum of the previous position and the new velocity by Eq.(7).

$$v_{ij}(t) = wv_{ij}(t-1) + c_1r_1(x_{ij}^{\#}(t-1) - x_{ij}(t-1)) + c_2r_2(x_j^*(t-1) - x_{ij}(t-1)) \quad (6)$$

$$x_{ij}(t) = x_{ij}(t-1) + v_{ij}(t) \quad (7)$$

Where r_1 and r_2 are the random numbers in the interval [0,1]. c_1 is a positive constant, called as coefficient of the self-recognition component, c_2 is a positive constant, called as coefficient of the social component. The variable w is called as the inertia factor, which value is typically setup to vary linearly from 1 to near 0 during the iterated processing. From Eq.(6), a particle decides where to move next, considering its own experience, which is the memory of its best past position, and the experience of its most successful particle in the swarm.

In the PSO model, the particle searches the solutions in the problem space within a range $[-s, s]$ (If the range is not symmetrical, it can be translated to the corresponding symmetrical range.) In order to guide the particles effectively in the search space, the maximum moving distance during one iteration is clamped in between the maximum velocity $[-v_{max}, v_{max}]$ given in Eq.(8), and similarly for its moving range given in Eq.(9):

$$v_{i,j} = sign(v_{i,j})min(|v_{i,j}|, v_{max}) \quad (8)$$

$$x_{i,j} = sign(x_{i,j})min(|x_{i,j}|, x_{max}) \quad (9)$$

The value of v_{max} is $\rho \times s$, with $0.1 \leq \rho \leq 1.0$ and is usually chosen to be s, i.e. $\rho = 1$. The pseudo-code for particle swarm optimization algorithm is illustrated in Algorithm 1.

Algorithm 1. Particle Swarm Optimization Algorithm

```
01. Initialize the size of the particle swarm $n$, and other parameters.
02. Initialize the positions and the velocities for all the particles randomly.
03. While (the end criterion is not met) do
04.   $t = t + 1$;
05.   Calculate the fitness value of each particle;
06.   $\mathbf{x}^* = argmin_{i=1}^{n}(f(\mathbf{x}^*(t-1)), f(\mathbf{x}_1(t)), f(\mathbf{x}_2(t)), \cdots, f(\mathbf{x}_i(t)), \cdots, f(\mathbf{x}_n(t)))$;
07.   For $i$= 1 to $n$
08.       $\mathbf{x}_i^{\#}(t) = argmin_{i=1}^{n}(f(\mathbf{x}_i^{\#}(t-1)), f(\mathbf{x}_i(t))$;
09.       For $j$ = 1 to $d$
10.           Update the $j$-th dimension value of $\mathbf{x}_i$ and $\mathbf{v}_i$
10.               according to Eqs.(6),(8),(7),(9);
12.       Next $j$
13.   Next $i$
14. End While.
```

4 A Fuzzy Particle Swarm Approach for QAP

For applying particle swarm algorithm successfully for an objective problem, one of the key issues is how to map the problem solution to the particle space, which directly affects its feasibility and performance. In a “crip” particle swarm model for the assignment problem, it would trend to assign many facilities to the same location or assign many locations to the same facility. This kind of the assignment would be unfeasible. In this section, a fuzzy matrix is introduced to represent the quadratic assignment problem. And then, a new approach to the problem space mapping is depicted for particle swarm optimization for the quadratic assignment problem.

Suppose $F = \{F_1, F_2, \cdots, F_n\}$, $L = \{L_1, L_2, \cdots, L_n\}$, then the fuzzy assignment relation from F to L can be expressed as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

Here a_{ij} represents the degree of membership of the j-th element F_j in domain F and the i-th element L_i in domain L to relation A. In the fuzzy relation matrix A between F and L, the elements subject to the following constraints:

$$a_{ij} = \mu_R(F_j, L_i), \quad i = 1, 2, \cdots, n, \quad j = 1, 2, \cdots, n. \tag{10}$$

μ_R is the membership function, the value of a_{ij} means the degree of membership that the facility F_j would be assigned to the location L_i in the feasible assignment solution. In the quadratic assignment problem, the elements of the solution must satisfy the following conditions:

$$a_{ij} \in \{0,1\}, \quad i=1,2,\cdots,n, \quad j=1,2,\cdots,n; \tag{11}$$

$$\sum_{i=1}^{n} a_{ij} = 1, \quad i=1,2,\cdots,n, \quad j=1,2,\cdots,n; \tag{12}$$

$$\sum_{j=1}^{n} a_{ij} = 1, \quad i=1,2,\cdots,n, \quad j=1,2,\cdots,n. \tag{13}$$

For applying PSO successfully, one of the key issues is how to map the problem solution to the particle space, which directly affects its feasibility and performance [11]. According to fuzzy matrix representation of the quadratic assignment problem, the position X and velocity V in the particle swarm are re-defined as follows:

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nn} \end{bmatrix}; \qquad V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{bmatrix}$$

The elements in the matrix X above have the same meaning as Eq.(10). Accordingly, the elements of the matrix X must satisfy the following conditions:

$$x_{ij} \in \{0,1\}, \quad i=1,2,\cdots,n, \quad j=1,2,\cdots,n; \tag{14}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \quad i=1,2,\cdots,n, \quad j=1,2,\cdots,n; \tag{15}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i=1,2,\cdots,n, \quad j=1,2,\cdots,n. \tag{16}$$

Because the position and velocity in the new fuzzy particle swarm model have been transformed to the form of matrices, they are updated by the new Eqs.(17) and (18) with the matrix operations.

$$\begin{aligned} V(t) = w \otimes V(t-1) \oplus (c_1 r_1) \otimes (X^{\#}(t-1) \ominus X(t-1)) \\ \oplus (c_2 r_2) \otimes (X^{*}(t-1) \ominus X(t-1)) \end{aligned} \tag{17}$$

$$X(t+1) = X(t-1) \oplus V(t) \tag{18}$$

The position matrix may violate the constraints (14), (15) and (16) after some iterations, it is necessary to normalize the position matrix. First we make all the negative elements in the matrix to become zero. If all elements in a column of the matrix are zero, they need be re-evaluated using a series of random numbers within the interval [0,1]. And then the matrix undergoes the following transformation:

$$Xnormal = \begin{bmatrix} x_{11}/\sum_{i=1}^{n} x_{i1} & x_{12}/\sum_{i=1}^{n} x_{i2} & \cdots & x_{1n}/\sum_{i=1}^{n} x_{in} \\ x_{21}/\sum_{i=1}^{n} x_{i1} & x_{22}/\sum_{i=1}^{n} x_{i2} & \cdots & x_{2n}/\sum_{i=1}^{n} x_{in} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1}/\sum_{i=1}^{n} x_{i1} & x_{n2}/\sum_{i=1}^{n} x_{i2} & \cdots & x_{nn}/\sum_{i=1}^{n} x_{in} \end{bmatrix}$$

Since the position matrix indicates the potential assigned solution, the fuzzy matrix can be "decoded" to the feasible solution. We choose the element which has the max value in the column, then tag it as "1", and other numbers in the column and row are set as "0" in the assigning matrix. After all the columns and rows have been processed, we get the assignment solution without violating the constraints (14), (15) and (16), and then calculate the assignment cost of the solution.

5 Experiment Settings, Results and Discussions

In our experiments, Ant Colony Optimization (ACO) was used to compare the performance with PSO. The two algorithms share many similarities. ACO deals with artificial systems that is inspired from the foraging behavior of real ants, which are used to solve discrete optimization problems [21]. The main idea is the indirect communication between the ants by means of chemical pheromone trials, which enables them to find short paths between their nest and food. It is implemented as a team of intelligent agents which simulate the ants behavior, walking around the graph representing the problem to solve using mechanisms of cooperation and adaptation. PSO is a stochastic search technique inspired by social behavior of bird flocking or fish schooling. Both methods are valid and efficient methods in numeric programming and have been employed in various fields due to their strong convergence properties. Specific parameter settings for the algorithms are described in Table 1. We consider the instances from Taillard's datasets[1] and QAPlib[2]. Each experiment (for each algorithm) was repeated 10 times with different random seeds. Each trial had a fixed number of $50 * n * n$ iterations (n is the dimension of the problem). If the value $50 * n * n$ is larger than $2 * 10^4$, the maximum iteration was set to $2 * 10^4$. The average costs (AvgCost) and the standard deviations (std) were calculated from the 10 different trials. The standard deviation indicates the differences in the results during the 10 different trials. Usually the main emphasis will be to generate the assignment solutions at a minimal amount of time. So the completion time for 10 trials were used as one of the criteria to improve their performance.

In order to closely track the performance of our algorithms, first we tested two small scale problems, nug5 and nug8. The nug5 is a simple QAP instance with 5 facilities on 5 locations. Its united matrix DF of the distance and flow is

[1] http://ina2.eivd.ch/collaborateurs/etd/
[2] http://www.opt.math.tu-graz.ac.at/qaplib/

Table 1. Parameter settings for the algorithms

Algorithm	Parameter name	Parameter value
ACO	Number of ants	5
	Weight of pheromone trail α	1
	Weight of heuristic information β	5
	Pheromone evaporation parameter ρ	0.8
	Constant for pheromone updating Q	10
PSO	Swarm size	5
	Self-recognition coefficient c_1	1.49
	Social coefficient c_2	1.49
	Inertia weight w	$0.9 \rightarrow 0.1$

$$\begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 5 & 2 & 4 & 1 \\ 2 & 1 & 0 & 3 & 0 & 2 \\ 3 & 1 & 2 & 0 & 0 & 0 \\ 4 & 2 & 1 & 1 & 0 & 5 \\ 5 & 3 & 2 & 2 & 1 & 0 \end{array}$$

Upper half of the DF matrix is the distance information, and the lower half is the flow information. The two algorithms both search the best value 50 in its 10 runs. The results for 10 ACO runs were all 50, while the results of 10 PSO runs were 50 nine times and 52 once. The optimal result is supposed to be 50 with the best permutation, (4,5,1,2,3). Figure 1 illustrates the performance curves during the search processes. ACO usually searches a better result using a less iteration number than PSO for the smaller scale problem.

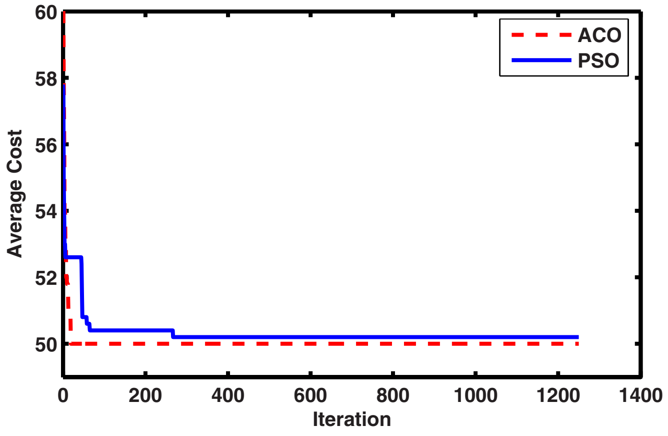

Fig. 1. Performance for nug5

The nug8 is other QAP instance with 8 facilities on 8 locations. The scale is a little larger than nug5. The united matrix DF of the distance and flow is

$$\begin{array}{c|cccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline 1 & 0 & 5 & 2 & 4 & 1 & 0 & 0 & 6 \\ 2 & 1 & 0 & 3 & 0 & 2 & 2 & 2 & 0 \\ 3 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 5 \\ 4 & 3 & 2 & 1 & 0 & 5 & 2 & 2 & 10 \\ 5 & 1 & 2 & 3 & 4 & 0 & 10 & 0 & 0 \\ 6 & 2 & 1 & 2 & 3 & 1 & 0 & 5 & 1 \\ 7 & 3 & 2 & 1 & 2 & 2 & 1 & 0 & 10 \\ 8 & 4 & 3 & 2 & 1 & 3 & 2 & 1 & 0 \end{array}$$

Figure 2 illustrates the performance of the two algorithms for nug8. The results for 10 ACO runs were {218,224,224,214,224,224,228,224,224,224}, with an average value of 222.8. The results of 10 PSO runs were {214,222,218,214,220,218, 218,218,222,224}, with an average value of 218.8. The optimal result is supposed to be 214 with the best permutation, (2,1,4,5,3,8,7,6). While ACO provided the best result once, PSO provided the best result twice.

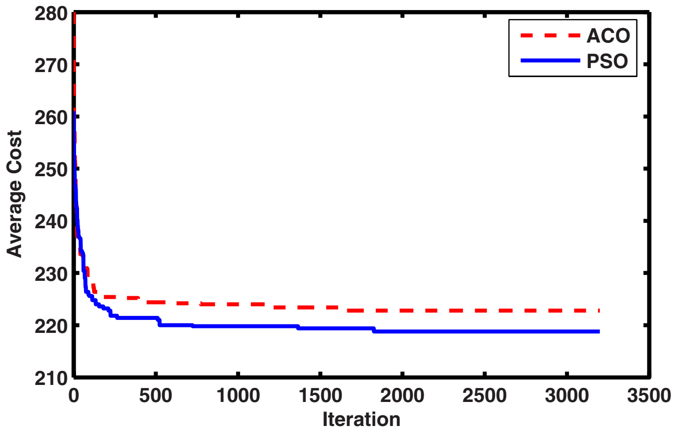

Fig. 2. Performance for nug8

Further, we tested the algorithms for other instances. The average cost (Avg-Cost), the standard deviations (std) and the time for 10 trials were recoded. Empirical results are summarized in Table 2. As the results depict, the ACO is an effective algorithm for the little scale problems, while PSO usually had better averages for a little bigger problem sizes. PSO also had larger standard deviations. The robustness of our algorithm is one of the future works in this study. It is to be noted that PSO usually spent the less time to assign the facilities on the locations.

Table 2. Comparing the results of ACO and PSO for quadratic assignment problems

Problem	ACO			PSO		
	AvgCost	std	time	AvgCost	std	time
nug5	50.0	0	258.4700	50.2	0.6325	103.8130
nug8a	222.8	3.9101	631.1193	218.8	3.2931	322.8750
tai8a	85934	800.4784	872.5578	83294	2698.1	572.0150
chr12a	16557	1661.6	1048.0	13715	2098.0	736.0545
tai12a	256180	3066.5	968.1560	254230	5809.9	653.1410
chr20a	5438.8	261.3909	1142.7	4456.0	389.8974	754.5
dre30	1849.6	82.1998	1514.4	1592.0	118.4736	1040.5
tho40	302840	3603.3	1612.3	286670	5318.3	1233.3
tai50a	5626356	15225	2045.3	5587622	52893	1602.4

6 Conclusions

In this paper, we introduced an approach based on Particle Swarm Optimization (PSO) for quadratic assignment problems. The representations of the position and velocity of the particles in PSO is extended from the real vectors to fuzzy matrices, through which we accomplished the mapping between the quadratic assignment problem and the particle. We evaluated the performance of our proposed approach and compared it with Ant Colony Optimization (ACO). Empirical results illustrated that the proposed approach was an effective approach to solve quadratic assignment problems and it outperformed ACO in the completion time.

Acknowledgements

The first author wishes to thank to Kyriakos Tsourapas, who provided the ACO source code for the compared experiments. This work is supported by NSFC (60373095) and MOE (KP0302).

References

1. Kennedy J, Eberhart R (2001) Swarm Intelligence. Morgan Kaufmann, CA
2. Clerc M (2006) Particle Swarm Optimization. ISTE Publishing Company, London
3. Eberhart R C, Shi Y (1998) Comparison between genetic algorithms and particle swarm optimization. In: Proceedings of IEEE International Conference on Evolutionary Computation, 611–616
4. Settles M, Rodebaugh B, Soule T (2003) Comparison of genetic algorithm and particle swarm optimizer when evolving a recurrent neural network, Cantu-Paz E, et al. (eds.): GECCO 2003, LNCS 2723, 148–149
5. Boeringer D W, Werner D H (2004) Particle swarm optimization versus genetic algorithms for phased array synthesis. IEEE Transactions on Antennas and Propagation, 52(3):771–779

6. Parsopoulos K E, Vrahatis M N (2002) Recent Approaches to Global Optimization Problems through Particle Swarm Optimization, Natural Computing 1:235–306
7. Schute J F, Groenwold A A (2005) A study of global optimization using particle swarms. Journal of Global Optimization, Kluwer Academic Publishers, 31:93–108
8. Ajith Abraham, He Guo and Hongbo Liu, Swarm Intelligence: Foundations, Perspectives and Applications, Swarm Intelligent Systems, Nadia Nedjah and Luiza Mourelle (Eds.), Studies in Computational Intelligence, Springer Verlag, Germany, pp. 3–25, 2006.
9. Kennedy J, Eberhart R C (1997) A discrete binary version of the particle swarm algorithm. in: Proceedings of International Conference on Systems, Man, and Cybernetics, IEEE Computer Society Press, 4104–4108
10. Shi X, Xing X, Wang Q, at el. (2004) A discrete PSO method for generalized TSP problem. in: Proceedings of International Conference on Machine Learning and Cybernetics, IEEE Computer Society Press, 2378–2383
11. Pang W, Wang K, Zhou C, Dong L (2004) Fuzzy discrete particle swarm optimization for solving traveling salesman problem. In: Proceedings of the Fourth International Conference on Computer and Information Technology, IEEE CS Press, 796–800
12. Ajith Abraham, Hongbo Liu, Weishi Zhang and Tae-Gu Chang, Job Scheduling on Computational Grids Using Fuzzy Particle Swarm Algorithm, 10th International Conference on Knowledge-Based & Intelligent Information & Engineering Systems, Springer Verlag, Germany, Lecture Notes in Artificial Intelligence, B. Gabrys, R.J. Howlett, and L.C. Jain (Eds.): Part II, Lecture Notes on Artificial Intelligence 4252, pp. 500-507, 2006.
13. Koopmans T C, Beckman M J (1957) Assignment problems and the location of economic activities. Econometrica, 25:53–76
14. Stützle T (2005) Iterated local search for the quadratic assignment problem. European Journal of Operational Research, In Press, Corrected Proof, ScienceDirect Available online 13 May 2005
15. Garey M R, Johnson D S (1979) Computers and Intractability: a Guide to the Theory of NP-Completeness. Freeman, CA
16. Angel E, Zissimopoulos V (2002) On the hardness of the quadratic assignment problem with metaheuristics. Journal of Heuristics, 8:399–414
17. Misevicivs A (2005) A tabu search algorithm for the quadratic assignment problem. Computational Optimization and Applications, 30:95–111
18. Maniezzo V, Colorni A (1999) The ant system applied to the quadratic assignment problem. IEEE Transactions on Knowlege and Data Engineering, 11(5):769–778
19. Drezner Z, Hann P M, Taillard E D (2005) Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for metaheuristic methods. Annals of Operations Research, 139:65–94
20. Gambardella L M, Taillard E D, Dorigo M (1999) Ant colonies for the quadratic assignment problem. Journal of the Operational Research Society, 50:167–176
21. Dorigo M, Stützle T (2004) Ant Colony Optimization, MIT Press, MA