Solving the Quadratic Assignment Problem with the Modified Hybrid PSO Algorithm

Ali Safari Mamaghani
Department of Computer Engineering
Bonab Branch
Islamic Azad University
Bonab, Iran
Ali.Safari.m@gmail.com

Mohammad Reza Meybodi
Department of Computer Engineering and Information Technology
AmirKabir University of Technology
Tehran, Iran
mmeybodi@aut.ac.ir

***Abstract*—In this paper a particle swarm optimization algorithm is presented to solve the Quadratic Assignment Problem, which is a NP-Complete problem and is one of the most interesting and challenging combinatorial optimization problems in existence. A heuristic rule, the Smallest Position Value (SPV) rule, is developed to enable the Continuous particle swarm optimization algorithm to be applied to the sequencing problems. So, we use SPV to the QAP problem, which is a discrete problem. A simple but very efficient Hill Climbing method is embedded in the particle swarm optimization algorithm.**

We test our hybrid algorithm on some of the benchmark instances of QAPLIB, a well-known library of QAP instances. This algorithm is compared with some strategies to solve the problem. The computational results show that the modified hybrid Particle Swarm algorithm is able to find the optimal and best-known solutions on instances of widely used benchmarks from the QAPLIB. In most of instances, the proposed method outperforms other approaches. Experimental results illustrate the effectiveness of proposed approach on the quadratic assignment problem.

***Index Terms*—The Quadratic Assignment Problem; Particular Swarm Optimization; Hill Climbing Approach; NP-Complete problem.**

I. INTRODUCTION

The Quadratic Assignment Problem (QAP) is one of the classical combinatorial optimization Problems and is widely regarded as one of the most difficult problem in this class. Given a set $N = \{1,2,3,...,n\}$ and $n \times n$ matrices $F = \{f_{ij}\}$ and $D = \{d_{ij}\}$, and $C = \{c_{ij}\}$, the QAP is to find a permutation ϕ of the set N which minimizes

$$z = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} d_{\phi(i)\phi(j)} + \sum_{i=1}^{n} c_{i\phi(i)}$$

As an application of the QAP, consider the following campus planning problem. On a campus, new facilities are to be erected and the objective is to minimize the total walking distances for students and staffs. Suppose there are *n* available sites and *n* facilities to locate. Let d_{kl} denotes the walking distance between the two sites *k* and *l* where the new facilities will be erected. Further, let $f_{i,j}$ denotes the number of people per week who travel between the facilities *i* and *j*. Then, the decision problem is to assign facilities to sites so that the walking distance of people is minimized. Each assignment can be mathematically described by a permutation ϕ of $N = \{1,2,3,...,n\}$ such that $\phi(i) = k$ means that the facility *i* is assigned to site *k*. The product $f_{ij} d_{\phi(i)\phi(j)}$ describes the weekly walking distance of people who travel between facilities i and j. Consequently, the problem of minimizing the total walking distance reduces to identifying a permutation ϕ that minimizes the function *z* defined above. This is an application of the QAP with each $c_{ik} = 0$. In this application, we have assumed that the cost of erecting a facility does not depend upon the site. In case it does, we will denote by $c_{i,k}$ the cost of erecting facility *i* at site *k*, and these costs will also play a role in determining the optimal assignment of facilities to sites [1].Additional applications of QAP include the allocation of plants to candidate locations, layout of plants, backboard wiring problem, design of control panels and typewriter keyboards, balancing turbine runners, ordering of interrelated data on a magnetic tape, processor-to-processor assignment in a distributed processing environment, placement problem in VLSI design, analyzing chemical reactions for organic compounds, and ranking of archaeological data.

On account of its diverse applications and the intrinsic difficulty of the problem, the QAP has been investigated extensively by the research community. The QAP has been proved to be an NP-complete problem and a variety of exact and heuristic algorithms have been proposed. Exact algorithms for QAP include approaches based on dynamic programming [2], cutting planes [3], and branch and bound [4, 5]. Among these, only branch and bound algorithms are guaranteed to obtain the optimum solution, but they are generally unable to solve problems of size larger than n=20. Since many applications of QAP give rise to problems of size far greater than 20, there is a special need for good heuristics for QAP that can solve large-size problems. Known heuristics for QAP can be classified into the following categories: construction methods [6, 7], limited enumeration methods [8, 9], local improvement methods [10], simulated annealing methods [11],

abu search methods [12, 13] and genetic algorithms [12, 14]. Among these, the tabu search method due to Skorin-Kapov [12] and the (randomized) local improvement method due to Li et al. [10] and Pardalos et al. [15], which they named as Greedy Randomized Adaptive Search Procedure (GRASP), are the two most accurate heuristic algorithms to solve the QAP. Other approaches were used to solve the QAP problem are ant colony algorithm [16, 17], Path Relinking method [18], hybrid GRASP approach with tabu search[19], hybrid Ant Colony approach with Genetic and Hill Climbing [20] and Parallel tabu search [21].

The structure of the reminder of this paper is as follows: Section 2 is an introduction to PSO algorithm. In section 3, we describe the proposed hybrid algorithm based on PSO and Hill Climbing to solve the problem. Section 4 and 5 are dedicated to describe experimental results and paper conclusion respectively.

II. Particular Swarm Optimization

Particle swarm optimization (PSO) is a population based computational technique inspired from the simulation of social behavior of flock of birds. PSO was originally designed and developed by Eberhart and Kennedy [22]. In PSO, a swarm of particle is used to represent the population of candidate solutions. Each particle is a point in the N-dimensional search space. A particle is represented by its current position, and its current velocity. PSO tries to find the optimal solution to the problem by moving the particles and evaluating the fitness of the new position.

III. Hybrid PSO Algorithm for the Quadratic Assignment Problem

Evaluation of each particle in the swarm requires the determination of the permutation of numbers 1...n since the value of function z in QAP problem is a result of the sequence. In this paper, we use a heuristic rule called Smallest Position Value (SPV) to enable the continuous PSO algorithm to be applied to all class of sequencing problems, which are NP-hard in the literature [23]. By using the SPV rule, the permutation can be determined through the position values of the particle so that the fitness value of the particle can then be computed with that permutation. Pseudo code of the PSO algorithm for the QAP is given in Figure 1.

```
Initialize parameters
Initialize population
Find sequence
Evaluate
Do {
    Find the personal best
    Find the global best
    Update velocity
    Update position
    Find sequence
    Evaluate
    Apply hill climbing (optional)
} While (Termination)
```

Fig. 1. The PSO Algorithm with Hill Climbing for the QAP Problem.

The basic elements of PSO algorithm is summarized as follows:

Particle: X_i^k denotes the i^{th} particle in the swarm at iteration k and is represented by n number of dimensions as $X_i^k = [x_{i1}^k,..,x_{in}^k]$, where x_{ij}^k is the position value of the i^{th} particle with respect to the j^{th} dimension ($j = 1,..,n$).

Population: pop^k is the set of m particles in the swarm at iteration k, i.e., $pop^k = [X_1^k,..,X_m^k]$.

Sequence: We introduce a new variable ϕ_i^k, which is a permutation of numbers 1...n implied by the particle X_i^k. It can be described as $\phi_i^k = [\phi_{i1}^k,..,\phi_{in}^k]$, where ϕ_{ij}^k is the assigned facility to location j of the particle i at iteration k with respect to the j^{th} dimension.

Particle velocity: V_i^k is the velocity of particle i at iteration k. It can be defined as $V_i^k = [v_{i1}^k,..,v_{in}^k]$, where v_{ij}^k is the velocity of particle i at iteration k with respect to the j^{th} dimension.

Inertia weight: w^k is a parameter to control the impact of the previous velocities on the current velocity.

Personal best: PB_i^k represents the best position of the particle i with the best fitness until iteration k. So, the best position associated with the best fitness value of the particle i obtained so far is called the *personal best*. For each particle in the swarm, PB_i^k can be determined and updated at each iteration k. In a minimization problem with the objective function $z(\phi_i^k)$ where ϕ_i^k is the corresponding sequence of particle x_i^k, the personal best PB_i^k of the i^{th} particle is obtained such that $z(\phi_i^k) \le z(\phi_i^{k-1})$ where ϕ_i^k is the corresponding sequence of personal best PB_i^k and ϕ_i^{k-1} is the corresponding sequence of personal best PB_i^{k-1}. To simplify, we denote the fitness function of the personal best as $z_i^{pb} = z(\phi_i^k)$. For each particle, the personal best is defined as $PB_i^k = [pb_{i1}^k,..,pb_{in}^k]$ Where pb_i^k is the position value of the i^{th} personal best with respect to the j^{th} dimension.

Global best: GB_i^k denotes the best position of the globally best particle achieved so far in the whole swarm. For this reason, the global best can be obtained such that $z(\phi^k) \le z(\phi_i^k)$ for i = 1,2,..,n where ϕ^k is the corresponding sequence of global best GB^k and ϕ_i^k is the corresponding sequence of particle best PB_i^k. To simplify, we denote the fitness function of the global best as $z^{pb} = z(\phi^k)$. The global best is then defined as $GB^k = [gb_1^k,..,gb_n^k]$ where gb_i^k is the position value of the global best with respect to the j[th] dimension.

978-1-4673-1740-5 /12/$31.00 ©2012 IEEE

Termination criterion: It is a condition that the search process will be terminated. It might be a maximum number of iteration or maximum CPU time to terminate the search.

Solution Representation: One of the most important issue when designing the PSO algorithm lies on its solution representation. In order to construct a direct relationship between the problem domain and the PSO particles for the QAP problem, we present n number of dimensions for n number of locations (j =1,.., n). In other words, each dimension represents a typical location. In addition, the particle $X_i^k = [x_{i1}^k,..,x_{in}^k]$ corresponds to the position values for n number of locations in the QAP problem. Each particle has a continuous set of position values. The particle itself does not present a sequence. Instead we use the SPV rule to determine the allocation of facilities to different locations implied by the position values x_{ij}^k of particle X_i^k.Table 1 illustrates the solution representation of particle X_i^k for the PSO algorithm with its velocity and corresponding sequence. According to the proposed SPV rule, the smallest position value is $X_{i5}^k = -1{,}20$, so the dimension j=5 is assigned to be the first facility $\phi_{i1}^k = 5$ in the sequence ϕ_i^k ; the second smallest position value is $X_{i2}^k = -0{,}99$, so the dimension *j*=2 is assigned to be the second facility $\phi_{i2}^k = 2$ in the sequence ϕ_i^k, and so on. In other words, dimensions are sorted according to the SPV rule, i.e., to x_{ij}^k values to construct the sequence ϕ_i^k. This representation is unique in terms of finding new solutions since positions of each particle are updated at each iteration *k* in the PSO algorithm, thus resulting in different sequences at each iteration *k*.

TABLE I. SOLUTION REPRESENTATION OF A PARTICLE

j	1	**2**	3	4	**5**	6
x_{ij}^k	1.80	**-0.99**	3.01	-0.72	-1.20	2.15
v_{ij}^k	3.89	2.94	3.08	-0.87	-0.20	3.16
ϕ_{ij}^k	5	**2**	4	1	6	3

In the above representation, the value of every cell (ϕ_{ij}^k) represents the facility that is assigned to corresponding location. In this example, there are 6 facilities to be placed at 6 locations. For example, the first cell means that facility 5 is placed at location 1; facility 2 is placed at location 2 and so on.

Initial population: A population of particles is constructed randomly for the PSO algorithm of the QAP problem. The continuous values of positions are established randomly. The following formula is used to construct the initial continuous position values of the particle: $x_{ij}^0 = x_{\min} + (x_{\max} - x_{\min}) \times U(0,1)$ where $x_{\min} = 0.0$, $x_{\max} = 4.0$ and U(0,1) a uniform random number between 0 and 1. Initial continuous velocities are generated by similar formula as follows: $v_{ij}^0 = v_{\min} + (v_{\max} - v_{\min}) \times U(0,1)$ where $v_{\min} = -4.0$, $v_{\max} = 4.0$ and U(0,1) a uniform random number between 0 and 1. Continuous velocity values are restricted to some range, namely $v_{ij}^k = [v_{\min}, v_{\max}] = [-4.0, 4.0]$ where $v_{\min} = -v_{\max}$.

As the formulation of the QAP problem suggests that the objective is to minimize cost Z, the fitness function value for the particle *i* of the population (or swarm) at the iteration k is,

$$Z_i^k(\phi_i^k) = \sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} d_{\phi^k(i)\phi^k(j)} + \sum_{i=1}^{n} c_{i\phi^k(i)}$$

Where ϕ_i^k is the corresponding sequence of particle x_i^k. For simplicity, $Z_i^k(\phi_i^k)$ will be denoted Z_i^k. The complete computational flow of the PSO algorithm for the QAP problem can be summarized as follows:

Step 1: Initialization

- Set *k*=0, *m*=size of swarm.
- *Generate m particles randomly as explained before,* $\{X_i^0, i = 1,..,m\}$ *where* $X_i^0 = [x_{i1}^0,..,x_{in}^0]$.
- *Generate initial velocities of particles randomly* $\{V_i^0, i = 1,..,m\}$ *where* $V_i^0 = [v_{i1}^0,..,v_{in}^0]$.
- *Apply the SPV rule to find the sequence* $\phi_i^0 = [\phi_{i1}^0,..,\phi_{in}^0]$ *of particle* X_i^0 *for i=1,.., m.*
- *Evaluate each particle i in the swarm using the objective function* Z_i^0 *for i=1,…, m.*
- *For each particle i in the swarm, set* $PB_i^0 = X_i^0$, *where* $PB_i^0 = [pb_{i1}^0 = x_{i1}^0,..., pb_{in}^0 = x_{in}^0]$ *along with its best fitness value,* $Z_i^{pb} = Z_i^o$ *for i=1,…, m.*
- *Find the best fitness value* $Z_l^0 = \min\{Z_i^o\}$ *for i=1,…, m with its corresponding position* X_l^0.
- *Set global best to* $GB^0 = X_l^0$ *where* $GB^0 = [gb_1 = x_{l1},..., gb_n = x_{ln}]$ *with its fitness value* $Z^{pb} = Z_l^o$

Step 2: Update iteration counter

- $k = k + 1$

Step3: Update inertia weight

- $w^k = w^{k-1} \times \alpha$ where α is decrement factor.

Step 4: Update velocity

- $v_{ij}^k = w^{k-1} v_{ij}^{k-1} + c_1 r_1 (pb_{ij}^{k-1} - x_{ij}^{k-1}) + c_2 r_2 (gb_j^{k-1} - x_{ij}^{k-1})$

Step 5: Update position

- $x_{ij}^k = x_{ij}^{k-1} + v_{ij}^k$

Step 6: Find Sequence

- Apply the SPV rule to find the sequence $\phi_i^k = [\phi_{i1}^k, .., \phi_{in}^k]$ for $i = 1, ..., m$

Step 7: Update personal best

- Each particle is evaluated by using its sequence to see if personal best will improve. That is, if $\phi_i^k < \phi_i^{pb}$ for $i = 1, ..., m$, then personal best is updated as $PB_i^k = X_i^k$ and $Z_i^{pb} = Z_i^k$ *for* $i = 1, ..., m$.

Step 8: Update global best

- Find the minimum value of personal best $Z_l^k = \min\{Z_i^{pb}\}$ *for* $i = 1, ..., m$, $l \in \{i : i = 1, ..., m\}$
- If $\phi_l^k < \phi^{gb}$, then the global best is updated as $GB^k = X_l^k$ and $Z^{gb} = Z_l^k$

Step 9: Stopping criterion

- If the number of iteration exceeds the maximum number of iteration, or maximum CPU time, then stop, otherwise go to step 2.

Hill climbing for the QAP problem: In the proposed PSO algorithm, Hill Climbing is applied to the sequence directly. However, it violates the SPV rule and needs a repair algorithm. This approach is illustrated in Table 2 and 3, where facility $\phi_{i2}^k = 2$ and facility $\phi_{i4}^k = 1$ are interchanged.

TABLE II. HILL CLIMBING APPLIED TO SEQUENCE BEFORE REPAIRING

j	**1**	**2**	**3**	**4**	**5**	6
x_{ij}^k	1.80	-0.99	3.01	-0.72	-1.20	2.15
Facility, ϕ_{ij}^k	5	**2**	4	**1**	6	3
x_{ij}^k	1.80	-0.99	3.01	-0.72	-1.20	2.15
Facility, ϕ_{ij}^k	5	**1**	4	**2**	6	3

As seen in Table 2, applying a Hill Climbing to the sequence violates the SPV rule because the sequence itself is a result of the particle's position values. Once a Hill Climbing is completed, particle should be repaired so that the SPV is not violated. This is achieved by changing the position values according to the SPV rule as shown in Table 3.

TABLE III. HILL CLIMBING APPLIED TO SEQUENCE AFTER REPAIRING

J	**1**	**2**	3	4	5	6
x_{ij}^k	**1.80**	**-0.99**	3.01	-0.72	-1.20	2.15
Facility, ϕ_{ij}^k	5	**2**	4	**1**	6	3
x_{ij}^k	**-0.99**	**1.80**	3.01	-0.72	-1.20	2.15
Facility, ϕ_{ij}^k	5	**1**	4	**2**	6	3

In other words, interchange the position values of the exchanged facilities in terms of their dimensions. Since facility $\phi_{i2}^k = 2$ and facility $\phi_{i4}^k = 1$ are interchanged, their associated position values $x_{i2}^k = -0.99$ and $x_{i1}^k = 1.80$ are interchanged for dimensions *j*=2 and *j*=1 to keep the particle consistent with the SPV rule.

The Pseudo Code of Hill Climbing for the QAP Problem is given in figure 2. In the figure p_{HC} is the probability of Hill climbing.

```
current= φ^k , sequence of global best GB^k ;
Iteration=0;
Do
{
   u=rnd(1,n);
   min_cost= int.MaxValue;
   for (i = 1; i <= n; i++)
      If (i!=u)
         {
              S=interchange(current, u,i);
              Repaire(s);
              If (Z(s)<min_cost)
                  {
                   min_cost=Z(s);
                   neighbour = S;
                  }
         }
   If (min_cost< Z(current)) current= neighbour;
} while (Iteration< p_HC * swarmsize);
```

Fig. 2. The Pseudo code of hill climbing for The QAP problem

IV. EXPERIMENTAL RESULTS

In this section the results of the implementation of the new algorithm is described. The hybrid PSO_{HC} algorithm presented for the QAP problem was coded in the C#.Net 2010 programming language.

We used the following parameters for the PSO. Social and cognitive parameters, and uniform random numbers are taken as c1 = c2 = 2 and r1= r2= 0.5 respectively. Initial inertia weight is set to w0 = 0.9 and never decreased below 0.40. Finally, the decrement factor α is taken as 0.975.We evaluated the performance of the proposed algorithm using the benchmark set of instances, available on the QAPLIB compiled by Burkard et al. [24]. (http://mscmga.ms.ic.ac.uk/jeb/orlib/wtinfo.html).

Experimental results and analysis are divided into two parts. First, in section A, we evaluate the effect of important parameters on the proposed algorithm. In section B, we present the computational results of the hybrid algorithm when applied to some instances of the QAPLIB.

A. Evaluate the effect of parameters on ICA algorithm

It is clear that the performance of the approximated algorithms is affected by parameter tuning. So, at first we do tuning process, to obtain good values for the key parameters.

We use bur26f instance of QAPLIB to test the effect of PSO_{HC} parameters. This instance contains 26 facilities. We increase the number of iterations of the algorithm ranging from 100 to 500. Figure 3 shows the results. The first point is that as the number of iterations increases, the quality of solutions will

978-1-4673-1740-5 /12/$31.00 ©2012 IEEE

mprove. There is another noticeable point in which the cost decreased sharply to 400 iterations, then it stabilizes. So, to balance the quality of the results and the running time, we set the maximum number of iterations to be 400 for following experiments.

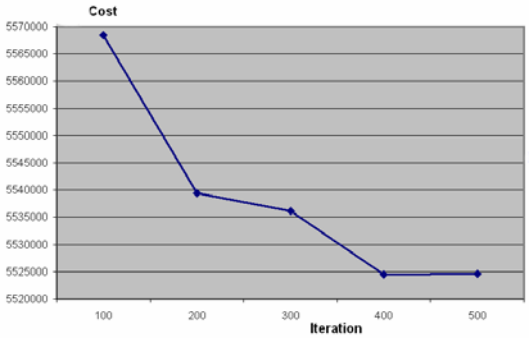

Fig. 3. The impact of increasing the PSO_{HC} 's number of iterations

We choose the size of swarm from the set {40, 80, 120, 160 and 200}. The figure 4 shows the results. The figure indicates that increasing the size of swarm makes better solutions. The quality of solutions improves until 120 particles, and then it is stable. So we've used 120 in our experiments.

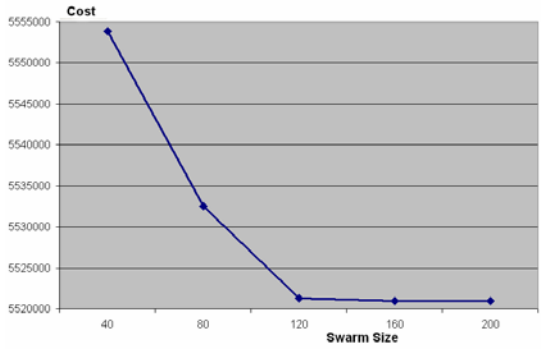

Fig. 4. The impact of increasing the size of swarm

B. Performance of Algorithm

Performance of the pure PSO, denoted as PSO_{pure}, and PSO with the Hill Climbing, denoted as PSO_{HC}, is compared with a simulated annealing method ,a kind of genetic algorithm which is called GA_OB and PSO with local Search, denoted as PSO_{VNS} [23].Table 5 shows the comparisons. The first column contains the QAP instance and second contains the size of problem. The third column shows the Best-Known Solution (BKS). The Solution and gap Column are respectively the solutions produced by each algorithm and the deviation in percentage from BKS.

According to the results, PSO_{HC} is able to find optimal and near optimal solutions. It means that in most cases the results are as well as BKS. In addition to the comparisons indicate that the new approach is better than other methods which have been proposed. So we can use PSO_{HC} approach as an effective method to solve the QAP problem.

V. CONCLUSIONS

The quadratic assignment problem is a very difficult combinatorial optimization problem. In order to obtain satisfactory results in a reasonable time, heuristic algorithms are to be applied. In this paper, we developed a hybrid modified PSO method in which a simple but very efficient Hill Climbing method is embedded.

By comparison the results produced by PSO_{HC} with the results by SA,GA, PSO_{pure} and PSO_{VNS} ,we find that PSO_{HC} do better and find the solutions which have low cost. These results are nearly as well as Best-Known Solutions. Experimental results illustrate the effectiveness of proposed approach on the quadratic assignment problem.

REFERENCES

[1] R.K. Ahuja, J.B. Orlin, and A. Tiwari," A greedy genetic algorithm for the quadratic assignment problem ", Computers and Operations Research 27, 917-934. 2000.

[2] Christo"des N, Benavent E. An exact algorithm for the quadratic assignment problem. Oper. Res. 1989;37:760-8.

[3] Bazara MS, Sherali MD. Benders' partitioning scheme applied to a new formulation of the quadratic assignment problem. Naval Res. Logist. Quart. 1980;27:29-41.

[4] Lawler EL. The quadratic assignment problem. Manag. Sci. 1963;9:586-99.

[5] Pardalos PM, Crouse J. A parallel algorithm for the quadratic assignment problem. Proceedings of the Supercomputing 1989 Conference, ACM Press. New York, 1989, pp. 351-60.

[6] Armour GC, Bu!a ES. Heuristic algorithm and simulation approach to relative location of facilities. Manag. Sci.1963;9:294 -309.

[7] Bula ES, Armour GC, Vollmann TE. Allocating facilities with CRAFT. Harvard Business Rev. 1962;42:136-58.

[8] West DH. Algorithm 608: approximate solution of the quadratic assignment problem.ACMTrans. Math. Software1983;9:461-6.

[9] Burkard RE, Bonniger T. A heuristic for quadratic boolean programs with applications to quadratic assignment problems. European J. Oper. Res. 1983;13:374-86.

[10] Li T, Pardalos PM, Resende MGC. A greedy randomized adaptive search procedure for the quadratic assignment problem. In: Pardalos PM, Wolkowicz H (Eds.), Quadratic Assignment and Related Problems. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. Providence. RI: Americal Mathematical Society, 1994, pp. 237-261.

[11] Wilhelm MR, Ward TL. Solving quadratic assignment problems by simulated annealing. IEEE Trans.1987;19:107-19.

[12] Skorin-Kapov J. Tabu search applied to the quadratic assignment problem. ORSA J. Comput. 1990;2:33-45. R.K. Ahuja et al. Computers & Operations Research 27 (2000) 917-934.

[13] Taillard E. Robust tabu search for the quadratic assignment problem. Parallel Comput. 1991;17:443-55.

[14] Fleurent C, Ferland JA. Genetic hybrids for the quadratic assignment problem. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 16, Providence. RI: American Mathematical Society, 1994, pp. 173-87.

[15] Resende MGC, Pardalos PM, Li Y. Fortran subroutines for approximate solution of dense quadratic assignment problems using GRASP. ACM Trans. Math. Software 1994;22:104-18.

[16] Stutzle, T., Dorigo, M., 1999. ACO algorithms for the quadratic assignment problem. In: Corne, D., Dorigo, M., Glover, F. (Eds.), New Ideas for Optimization. McGraw-Hill, pp. 33–50.

[17] Gambardella, L., Taillard, E., Dorigo, M., 1997. Ant Colonies for the QAP, Tech. Report IDSIA-4-97, IDSIA, Lugano, Switzerland.

[18] James, T., Rego, C., Glover, F., 2005. Sequential and parallel pathrelinking algorithms for the quadratic assignment problem. IEEE Intelligent Systems 20 (4), 58–65.

[19] Oliveira, C.A., Pardalos, P.M., Resende, M.G.C., 2004. GRASP with pathrelinking for the quadratic assignment problem, Efficient and Experimental Algorithms. In: Ribeiro, C.C., Martins, S.L. (Eds.), . In: Lecture Notes in Computer Science, vol. 3059. Springer-Verlag, pp. 356–368.

[20] Tseng, L., Rego, C., Glover, F.,2009. A cooperative parallel tabu search algorithm for the quadratic assignment problem. European Journal of Operational Research 195, 810–826.

[21] James, T., Liang, S., 2005. A hybrid metaheuristic for th quadratic assignment problem. Computational Optimization and Applications 34, 85–113.

[22] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization," in IEEE International Conference on Neural Networks, Piscataway, NJ, 1995, pp. 1942-1948.

[23] M. Fatih Tasgetiren, Mehmet Sevkli, Yun-Chia Liang, Gunes Gencyilmaz, "Particle Swarm Optimization Algorithm for the Single Machine Total Weighted Tardiness Problem", In: the Proceeding of the World Congress on Evolutionary Computation (CEC2004), 2004, p.1412-1419.

[24] Burkard RE, Karisch SE, Rendl F. QAPLIB - A quadratic assignment program library. J. Global Optim.1997;10:391-403.

TABLE IV. PERFORMANCE COMPARISON ACCORDING TO PROBLEM SIZE

			PSO_{Pure}		PSO_{VNS}		PSO_{HC}		SA		OB-GA	
Problem	**N**	**BKS**	**Solution**	**Gap(%)**	**Solution**	**Gap(%)**	**Solution**	**Gap(%)**	**Solution**	**Gap(%)**	**Solution**	**Gap(%)**
tai10a	10	135028	135028	0.000	135028	0.000	135028	0.000	135028	0.000	135028	0.000
tai20a	20	703482	704190	0.101	703482	0.000	703482	0.000	703482	0.000	703482	0.000
tai30a	30	1818146	1868871	2.790	1866122	2.639	1848571	1.673	1849696	1.735	1863722	2.506
tai40a	40	3139370	3233491	2.998	3225382	2.740	3211400	2.294	3212692	2.335	3215382	2.42125012
tai50a	50	4941419	5088574	2.978	5043551	2.067	5040012	1.995	5041058	2.016	5067904	2.559
tai60a	60	7208572	7438398	3.188	7431459	3.092	7325133	1.617	7381442	2.398	7422318	2.965
tai80a	80	13557864	13831255	2.016	13812280	1.877	13800423	1.789	13852100	2.170	13814562	1.893
tai100a	100	21125314	21603991	2.266	21602881	2.267	21483465	1.695	21499478	1.771	21501554	1.780
tai150b	150	498896643	508956543	2.016	506448890	1.514	502349984	0.692	502651565	0.572	506448890	1.513
tai256c	256	44759294	44819486	0.134	44809949	0.113	44799323	0.089	44814014	0.122	45023121	0.589
sko100a	100	152002	155462	2.276	153949	1.281	152922	0.605	154210	1.452	153090	0.715
sko100b	100	153890	156007	1.376	155971	1.352	154123	0.151	154262	0.241	155030	0.740
sko100c	100	147862	150978	2.107	149366	1.017	149742	1.271	149542	1.136	149948	1.410
sko100d	100	149576	152346	1.852	151746	1.451	150431	0.572	151746	1.450	150828	0.837
sko100e	100	149150	152349	2.145	150999	1.240	151426	1.526	150426	0.855	150598	0.970
sko100f	100	149036	152038	2.014	150871	1.231	150111	0.721	150738	1.142	150402	0.916

978-1-4673-1740-5 /12/$31.00 ©2012 IEEE