International Journal of Industrial Engineering Computations 9 (2018) 491–508

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Development of modified discrete particle swarm optimization algorithm for quadratic assignment problems

T.G. Pradeepmon, R. Sridharan and Vinay V. Panicker*

Department of Mechanical Engineering, National Institute of Technology, Calicut, India

C H R O N I C L E

Article history:
Received August 18 2017
Received in Revised Format August 25 2017
Accepted November 15 2017
Available online November 15 2017

Keywords:
Discrete Particle Swarm Optimization
Quadratic Assignment problem

A B S T R A C T

Particle swarm optimization has been established to be one of the efficient algorithms for finding solutions for continuous optimization problems. The discretized form of particle swarm optimization, known as the discrete particle swarm optimization is an efficient tool for solving combinatorial optimization problems and other problems involving discrete variables. In this paper, a revised version of the discrete particle swarm optimization algorithm is proposed for solving Quadratic Assignment Problems (QAP). Instead of using the general velocity and position update procedures in particle swarm optimization algorithms, four different possible positions are found out for each particle and the best among them is accepted as the updated position. The algorithm is applied to solve some benchmark instances of QAP taken from QAP Library and the results show minute deviations from best-known solutions.

1. Introduction

The Quadratic Assignment Problem (QAP) first appeared in literature during 1957 when Koopmans and Beckman published a paper on the allocation of indivisible resources. The paper discussed two problems related to the location of economic activities which was interpreted as problems of assigning plants to locations. Out of the two problems discussed, the first one ignored the cost of transportation between plants and found to be a linear programming problem. The second problem considered the interplant transportation hence formulated as a quadratic assignment problem (Koopmans & Beckmann, 1957). QAP aims to decide on the best placement of a number of facilities (say n) to an equal number (n) of locations such that the total cost is minimised. The total cost includes the cost of placing facilities in the respective locations and the cost of transporting materials between the facilities. Consider three matrices,

1. $A = (a_{ik})$, the flow of material between the facilities, where a_{ik} is the flow from facility i to facility k for all $i, k \in \{1, \ldots, n\}$,
2. $B = (b_{jl})$, the distance between the locations, where b_{jl} is the distance between locations j and l for all $j, l \in \{1, \ldots, n\}$, and

* Corresponding author
E-mail: vinay@nitc.ac.in (V. V. Panicker)

2018 Growing Science Ltd.
doi: 10.5267/j.ijiec.2017.11.003

3. $C = (c_{ij})$, the cost of placing facilities in locations, where c_{ij} is the cost of locating facility i in location j for all $i, j \in \{1, \ldots, n\}$,

The quadratic assignment problem can be defined as to find the permutation π of n facilities so as to minimize

$$Z = \sum_{i=1}^{n} c_{i\pi(i)} + \sum_{i=1}^{n}\sum_{k=1}^{n} a_{ik} b_{\pi(i)\pi(k)}$$

Since its introduction, the QAP is celebrated for its capability to represent a variety of real-world applications such as plant layout, backboard wiring on electrical circuits, placement algorithms in VLSI design, design of control panels and keyboards, hospital and campus layout planning, ordering of runners in a relay race team, ranking of archaeological data, the analysis of chemical reactions and many more (Burkard et al., 2009). Many classical combinatorial optimization problems including travelling salesperson problem, maximum clique problem, and graph partitioning problem etc. can also be expressed as QAP. Detailed reviews on QAP with application areas and solution methodologies can be found in Zaied and Shawky (2014) and Loila et al. (2004).

The QAP is proved to be NP-Hard (Garey & Johnson, 1979) in nature and because of its complexity, there may not be any polynomial time algorithm that can solve the problem. Even small instances of QAP require considerably large computational time. Obtaining an approximate solution for QAP with a definite performance is also proved to be very hard (Hassin et al., 2009). The main exact algorithm approaches for solving QAP are based on a branch-and-bound algorithm, dynamic programming and cutting plane algorithm. Out of these three methods, only branch-and-bound algorithms guarantee an optimum solution, that too for problems of size less than $n = 30$ (Loiola et al., 2007). Thus, the heuristic and metaheuristic methods become the natural choice of researchers solving QAPs. These algorithms provide near-optimal solutions within acceptable computational time. The set of benchmark instances provided by various researchers are used for assessing the performance of these algorithms. Among the numerous heuristic and metaheuristics algorithms reported in the literature, Genetic Algorithms (GAs) (Ahmed, 2015a), Simulated Annealing (Misevicius, 2003), Tabu Search (TS) (Czapiński, 2013), Ant Colony Optimization (ACO) (Hong, 2013), Neural Networks (Uwate et al., 2004), and Iterated Local Search (Ramkumar et al., 2008) are some of the familiar algorithms that have been successful in solving QAP, at least to a near optimal solution.

The ability of the searching mechanism in exploring the solution space is important in finding global solutions of optimization problems. The particle swarm optimization (PSO) algorithm proceeds by updating the position of particles learning from its inertia, a personal best position attained, local best particle and global best particle. Thus, PSO is good at searching the solution space globally and locally. The general PSO is suitable to continuous optimization problems and for resolving combinatorial optimization problems like QAP, a discrete version of PSO is needed.

A revised version of the Discrete Particle Swarm Optimization algorithm (DPSO) for resolving QAP is presented in this paper. Different parameters and operators used in the proposed DPSO are adopted from Pradeepmon et al. (2016), in which Taguchi's design of experiments method is used for finding the best combination of parameters and operators for the algorithm. Benchmark instances from QAPLIB are used during the tests. The DPSO provides good near-optimal solutions for the benchmark instances considered.

The remainder of the paper is organised as follows: In Section 2, a summary of the solution methodologies of QAP is given. It outlines the exact as well as heuristic and metaheuristic solution methods for QAP. In Section 3, the working of classical PSO is explained. Section 4 describes the proposed algorithm in detail and in Section 5 various parameters and different operators used in DPSO are explained. Section 6 gives a short illustration of the proposed algorithm considering a small problem. Section 7 presents the results and related discussions. Finally, Section 8 provides concluding remarks and areas for further research.

Notations

X_i – position vector of i^{th} particle
x_{ij} – j^{th} element in position vector of i^{th} particle
n – number of decision parameters or number of elements in a vector
V_i – velocity vector of i^{th} particle
v_{ij} – j^{th} element in velocity vector of i^{th} particle
$Pbest_i$ – personal best position vector of i^{th} particle
p_{ij} – j^{th} element in personal best position vector of i^{th} particle
m – number of particles in a swarm
$Gbest$ – global best position
N – the problem size
t – iteration number
T_{max} – maximum number of iterations
w – inertia weight ranging between [0, 1]
c_1 and c_2 – cognitive and social learning factors, respectively
r_1 and r_2 – random numbers ranging between [0, 1]
Π^t – swarm during iteration number t
Π_i^t – i^{th} particle in the swarm during iteration number t
λ_i, ε_i, δ_i, γ_i – inertia, cognition, socio-local and socio-global components of velocity respectively of i^{th} particle
$X_{i(\lambda)}$, $X_{i(\varepsilon)}$, $X_{i(\varepsilon)}$, $X_{i(\varepsilon)}$ – position of i^{th} particle considering only inertia, cognition, socio-local and socio-global components of velocity respectively
μ – mutation probability
η_1, η_2 and η_3 – crossover probabilities associated with cognition, socio-local and socio-global crossover operations
Θ – operator for incorporating mutation and crossover probabilities.
Θ – mutation operator for finding position due to inertia
Φ_1, Φ_2 and Φ_3 – crossover operators for finding position due to cognition, socio-local and socio-global components of velocity respectively
$F(x)$ – is the fitness function

2. Literature Survey

Since its introduction in 1957 by Koopmans and Beckmann, the QAP has been exploited in a variety of situations such as Facility Layout Problem, Campus Planning, University Examination Scheduling, Hospital Layout, Turbine Balancing, Typewriter Keyboard Design, Computer Manufacturing, Printed Circuit Board (PCB) Assembly, Room Allocation Problem, etc. Its diverse applications and complex nature made it one of the favourite problems of researchers in the field of operations management. The QAP is a proven NP-Hard problem and no polynomial time algorithm is currently available which can solve the problem optimally. This made heuristic and metaheuristic methods popular among researchers working on QAP. A number of exact, heuristic and metaheuristic algorithms providing exact and near-optimal solutions of QAP are available in the literature. The exact algorithms are guaranteed to provide optimal solutions but are limited to solving only small sized problems. The exact methods used for solving QAPs are Branch-and-Bound (Brixius & Anstreicher, 2000; Clausen & Perregaard, 1997; Hahn et al., 2001), Cutting plane algorithms (Bazaraa & Sherali, 1982) and Dynamic Programming methods

(Urban, 1998). The heuristic algorithms guarantee near-optimal solutions within a short period. But, as the problem size increases, the gap between the obtained solution and the optimal solution also increases (Osman & Laporte, 1996; Xia, 2010). The various categories of heuristic methods are construction methods (Arkin et al., 2001; Fleurent & Glover, 1999), limited enumeration methods (West, 1983) and improvement methods (Anderson, 1996; Li & Smith, 1995; Misevicius, 2000). The optimal solution is not guaranteed by using metaheuristic methods, but it promises a near-optimal solution within a short time, irrespective of the problem size. It is also possible that the obtained solution is the optimal one.

Metaheuristic methods are general purpose generic iterative procedures which guide a heuristic search toward promising regions in the search space of an optimization problem. They can be implemented for solving a wide variety of problems with minor modifications to customise them for a particular problem. These methods are generally classified into single solution methods and population based methods. The metaheuristic methods include Genetic Algorithm (GA), Simulated Annealing, Ant Colony Optimization (ACO), Tabu Search, etc. and many hybrid algorithms. Some works have been reported, which employ GA and its variants for solving QAPs. By using simple GA good results can be found for small instances of QAP as reported by Tate and Smith (Tate & Smith, 1995). But for larger problems of size above 20, simple GA is not able to obtain best-known solutions. To overcome this shortcoming a number of researchers hybridised GA with other methods to obtain good solutions for higher sized instances (Ahmed, 2015b; Drezner & MisevicIus, 2013; Drezner, 2008; Misevicius & Guogis, 2012; Misevicius, 2004). There are a variety of GA variants available in the literature which has been applied for solving QAPs (Ahmed, 2015a; Azarbonyad & Babazadeh, 2014; Day et al., 2003; Tosun, 2014; Wu & Ji, 2007).

Burkard and Rendl (1984) were the first to implement Simulated Annealing for solving QAPs and Connolly (1990) refined it. Further applications of Simulated Annealing for solving QAPs can be found in (Paul, 2011; Peng et al., 1996; Wilhelm & Ward, 1987). Parallel implementations of Simulated Annealing for improving the performance (in some cases up to 50-100 times better performance can be obtained by parallelization) can be found in (Paul, 2012). Performance comparison of Simulated Annealing with Tabu Search can be found in (Battiti & Tecchiolli, 1994) and it is argued that Simulated Annealing performs better for a comparatively lower number of iterations. But, when the problem is of higher complexity, the number of iterations needed increases, and in that case, Tabu Search outperforms Simulated Annealing. The first implementation of Tabu Search for solving QAP was performed by Skorin-Kapov (1990). Tabu Searches is the main candidate for the parallelization of algorithms and hardware implementations for solving QAPs (Czapiński, 2013; Matsui et al., 2004; Talbi et al., 1998; Wakabayashi et al., 2006; Zhu et al., 2010). Drezner (2005) extended the concentric tabu search by including more number of possible moves for cracking the QAP. Two extensions are suggested and tested. James et al. (2009) presented a cooperative parallel tabu search algorithm (CPTS) in which the information exchange takes place between processors throughout the run of the algorithm.

Gambardella et al. (1999) proposed the first ACO implementation for QAP, in which the ant colony system is hybridised with a local search (HAS-QAP). See and Wong (2008) provided a broad review of the notions of ACO, its uses and various ACO algorithms or variants developed for solving QAPs. There are a number of hybrid metaheuristics and variants of simple algorithms available in the literature. Tseng and Liang (2006) proposed a hybrid metaheuristic combining the ACO, the GA and a local search method and the method is called ANGEL. The ANGEL is marked by its two key phases - an ACO phase and a GA phase. A large number of QAP benchmark instances were tried and the results confirm that the proposed algorithm is competent enough to achieve the optimal solution with good efficiency. The Neural Meta-Memes Framework proposed by Song et al. (2011) is a combination of different algorithms namely Genetic Algorithm, Simulated Annealing, Tabu Search, and Iterated Local Search. The proposed framework was applied on QAPs with success.

Even though there are a large number of publications on QAP in the last few decades, not many of them are adopting Particle Swarm Optimization (PSO) as solution methodology. This may be because of the fact that PSO is mainly used for solving continuous optimization problems. The idea of learning in PSO is realized by using the notion of Euclidean distance between different solutions. Euclidian distance

cannot be calculated for combinatorial optimization problems. QAP is a well-known combinatorial optimization problem and the biggest challenge for the application of PSO to QAP is to establish an appropriate distance measure. There are some papers which use PSO for hybridising with other methods. Liu and Abraham (2007) combined fuzzy variable neighbourhood search with PSO and compared the algorithm with other four algorithms in terms of its performance in solving only a single problem taken from QAP Library. Mamaghani and Meybodi (2012) hybridised Hill Climbing Approach with PSO and the performance of the suggested algorithm is matched with other algorithms. The results are promising with variation from best-known solution as low as one percent. Hong (2013) hybridized ACO with PSO and compared the results with that obtained for simple ACO for four problems taken from QAP Library. Hafiz and Abdennour (2016) proposed a probability-based approach and based on this concept, a generic framework for discretizing PSO is developed. Based on this framework, five PSO variants are discretized and applied on QAP. There are some hardware implementations of PSO for solving QAP. Szwed et al. (2015) proposed a PSO algorithm and Szwed and Chmiel (2015) proposed a multi-swarm PSO algorithm for the QAP to be implemented on OpenCL platform. The parallel implementations of these algorithms performed better than sequential implementations on low-end devices.

All the available implementations of PSO for solving QAPs are either hybrid methods or are applicable only to a small set of problems. In this research, we propose a variant of the general PSO applied for permutations problems, in which, instead of adopting the commonly used velocity and position update procedures, explores four different possible positions for each particle and the best among them is accepted as the updated position. This updating procedure enhances the convergence of the algorithm. Further, the optimal parameter setting of DPSO is determined using Taguchi's design of experiments method (Pradeepmon et al., 2016). The suggested DPSO is applied to solve a number of benchmark problems taken from QAP library.

3. Particle Swarm Optimization

Kennedy and Eberhart (1995) introduced Particle Swarm Optimization (PSO) as a method for optimizing continuous nonlinear functions. It is derived from the social-psychological theory and simulates the social behaviour of bird swarming and fish schooling. The PSO is a population based search method, like Genetic Algorithm (GA), but does not use operations like mutation and crossover. The PSO retains a swarm (population) of particles (representing possible solutions). Each particle is described by a group of three vectors denoted as

$$(X_i, V_i, Pbest_i)$$

where $X_i = \{x_{i1}, x_{i2}, \ldots, x_{in}\}$ $(i = 1, 2, \ldots, m)$, $V_i = \{v_{i1}, v_{i2}, \ldots, v_{in}\}$ $(i = 1, 2, \ldots, m)$ and $Pbest_i = \{p_{i1}, p_{i2}, \ldots, p_{in}\}$ $(i = 1, 2, \ldots, m)$ are vectors representing the position, velocity and personal best position attained for the i^{th} particle in a swarm with m particles. The particles fly through the search space influenced by its own velocity (inertia), the best position attained by itself (personal best; *Pbest*) and the best position attained by the whole swarm (global best; $Gbest = \{g_1, g_2, \ldots, g_n\}$). The movement of particles in the search space is governed by the velocity and position updating equations as given below:

$$v_{ij}^{t+1} = wv_{ij}^{t} + c_1 r_1 \left(p_{ij}^{t} - x_{ij}^{t} \right) + c_2 r_2 \left(g_j^{t} - x_{ij}^{t} \right)$$

$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1}$$

where t represents the iteration number, w is the inertia weight (coefficient to share the knowledge represented in the previous velocities with the current velocity) ranging between 0 and 1; c_1 and c_2 are called cognitive and social learning factors, respectively; and r_1 and r_2 are random numbers ranging between 0 and 1.

During each iteration, the velocity and position of the particles in the swarm are updated using the velocity and position updating procedures, and the new personal best (*Pbest*) and global best (*Gbest*)

values and corresponding particles are identified. The *Pbest* of each particle in the swarm is modified using the following criterion.

$$Pbest_i^t = \begin{cases} X_i^t & \text{if } f(X_i^t) \geq f(Pbest_i^{t-1}) \\ Pbest_i^{t-1} & \text{if } f(X_i^t) < f(Pbest_i^{t-1}) \end{cases} \qquad 1 \leq i \leq m$$

The current *Gbest* of the swarm is updated as follows:

$$Gbest^t = \begin{cases} X_i^t & \text{if } f(X_i^t) \geq f(Gbest^{t-1}) \quad \forall\, i \\ Gbest^{t-1} & \text{if } f(X_i^t) < f(Gbest^{t-1}) \quad \forall\, i \end{cases} \qquad 1 \leq i \leq m$$

The original PSO algorithms are used for optimizing problems in which the elements of the solution are continuous real numbers. The PSO has been applied successfully for solving a variety of problems involving optimization, such as system design, multi-objective optimization (Peng et al., 2013), pattern recognition, medical field (Asarry et al., 2013, De et al., 2012), signal processing, games, robotic applications, decision making etc. (Eberhart & Shi, 2001). More details on PSO can be obtained from Zhang et al. (2015).

Even though the initial implementation of PSO was for continuous optimization problems, later many discrete adaptations of PSO, known as Discrete Particle Swarm Optimization (DPSO) were used for solving discrete optimization problems such as the minimum labelling Steiner tree problem (Consoli et al., 2010), Scheduling (Izakian et al., 2010; Lian et al., 2014), warehouse location problem (Ozsoydan & Sarac, 2011), Sensor Deployment Problem (Rapai et al., 2008) and p-median problem (Sevkli, 2014). Various discretization methods for PSO were proposed by Tasgetiren et al. (2006) and Pan et al. (2008).

4. Proposed Discrete Particle Swarm Optimization

In this study, a unique DPSO algorithm is proposed for the QAP. In this novel approach, the separate velocity vector is avoided, a socio-local component of velocity is also considered for position update of each particle and the position updating procedure finds four different possible movements of the particle and the one selected using rank selection method is selected as the updated position.

4.1. Particle representation

The widely accepted permutation representation of QAP is used in this study and thus, each particle is denoted as a permutation of *N* integers, where *N* is the problem size. For example, in a QAP with five facilities (i.e., $N = 5$), a feasible solution represented by the permutation {3 4 5 1 2} indicates that third facility is allotted to the location number one, the fourth facility to location number two and so on. Thus, the position of each particle is a permutation of *N*.

4.2. Swarm initialization

In the general PSO, the swarm is represented as $\Pi^t = [\Pi_1^t, \Pi_2^t, ..., \Pi_m^t]$, where *m* is the number of particles in the swarm (swarm size) and Π_i^t is the i^{th} particle in the swarm Π^t during iteration *t* and $\Pi_i^t = [X_i^t, V_i^t, Pbest_i^t]$. An initial swarm of random particles with velocity and position is generated and fitness value of each particle is calculated. The initial position of each particle is assigned as its *Pbest* and position of the particle with best fitness value among all particles in the swarm is assigned as the *Gbest*. In the proposed algorithm, the velocity vector is avoided and each particle holds only its position and its personal best, i.e., $\Pi_i^t = [X_i^t, Pbest_i^t]$. Instead of using a velocity vector for a mutation operator is used for updating the position of the particle based on its inertia. Or in other words, the position vector itself acts as the velocity vector. This reduces the memory space requirement for execution of the algorithm.

4.3. *Velocity and Position updating*

In every iteration, the particles belonging to the swarm move around in the solution space searching for better positions. The movement of particles to the new positions is influenced by (i) current position, (ii) velocity, (iii) best position so far attained by the particle, (iv) best position so far attained by particles in the neighbourhood and (v) the global best position attained by the entire swarm. Thus, the velocity is influenced by four components namely, inertia component (λ_i^t), cognition component (ε_i^t), socio-local component (δ_l^t) and socio-global component (γ^t). In some variants of PSO, the socio-local component of velocity is omitted. Figure 1 represents the methodology for updating particle position in classic PSO. Generally, the velocity and position of the particle will get updated using following the following equations:

$$V_i^{t+1} = \left[w \otimes F_1\left(\lambda_i^t\right)\right] \oplus \left[c_1 \otimes F_2\left(\varepsilon_i^t\right)\right] \oplus \left[c_2 \otimes F_3\left(\delta_l^t\right)\right] \oplus \left[c_3 \otimes F_4\left(\gamma^t\right)\right]$$

$$X_i^{t+1} = F_5\left(X_i^t, V_i^{t+1}\right)$$

where, F_1, F_2, F_3, F_4 and F_5 are the operators feasible in a permutation space, w is the weight given to inertia, c_1, c_2 and c_3 are the learning factors for cognition, socio-local and socio-global knowledge.

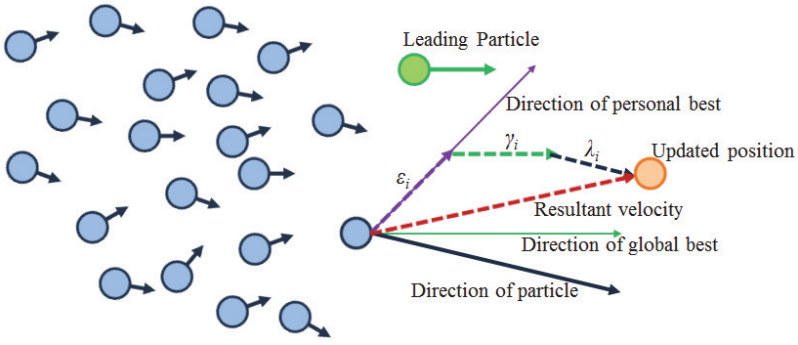

Fig. 1. Position updating strategy for classic PSO

In the proposed DPSO algorithm, a novel position updating strategy is proposed in which four possible movements of the particle incorporating four components of velocity taking one at a time are considered. Thus, there will be four possible positions for the i^{th} particle based on its own inertia ($X_{i(\lambda)}$), cognition (personal best) ($X_{i(\varepsilon)}$), local best particle ($X_{i(\delta)}$) and global best particle ($X_{i(\gamma)}$). Out of these four possible positions, the one with maximum fitness is selected as the updated position for the particle. Mutation operations used in Genetic Algorithm are used for finding the position based on inertia and for other positions crossover operators are used. Figure 2 represents the proposed position updating methodology used in the DPSO algorithm.

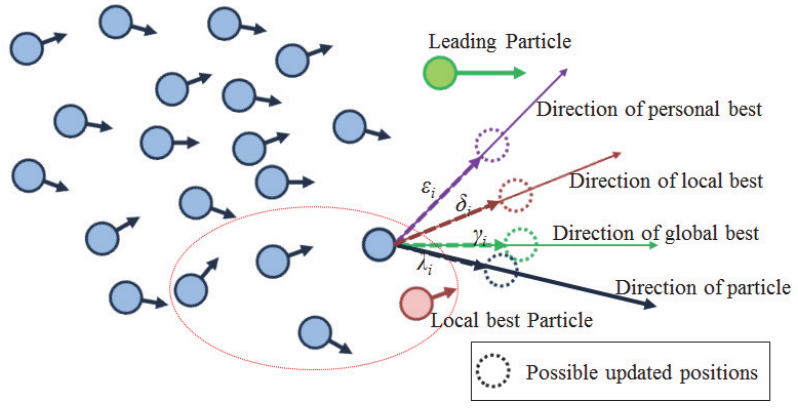

Fig. 2. Position updating strategy for proposed DPSO

The four potential positions and the position selected for movement of the particle are obtained as follows:

$$X_{i(\lambda)}^{t} = \mu \odot \Theta\left(X_i^t\right)$$

$$X_{i(\varepsilon)}^{t} = \eta_1 \odot \Phi_1\left(X_i^t, Pbest_i^t\right)$$

$$X_{i(\delta)}^{t} = \eta_2 \odot \Phi_2\left(X_i^t, Lbest_l^t\right)$$

$$X_{i(\gamma)}^{t} = \eta_3 \odot \Phi_3\left(X_i^t, Gbest^t\right)$$

$$X_i^{t+1} = \arg\max F(X) \text{ where } X = \left\{X_{i(\lambda)}^t, X_{i(\varepsilon)}^t, X_{i(\delta)}^t, X_{i(\gamma)}^t\right\}$$

μ is the mutation probability, η_1, η_2 and η_3 are the crossover probabilities associated with cognition, socio-local and socio-global crossover operations, Θ is the mutation operator for finding position due to inertia, Φ_1, Φ_2 and Φ_3 are the crossover operators for finding position due to cognition, socio-local and socio-global components of velocity respectively and $F(x)$ is the fitness function. The combination of operators and parameters for the proposed DPSO is borrowed from the work of Pradeepmon et al. (2016), in which the Taguchi's robust design method is applied to finding the optimised parameter setting for DPSO used for solving QAPs.

5. Parameters and Operations in DPSO

The parameters to be decided in the DPSO are mutation probability, crossover probabilities for three different crossovers, namely, Cognition Crossover, Socio-local Crossover and Socio-Global Crossover, and the swarm size. The various operations involved in DPSO are position updating strategy, mutation, cognition crossover, socio-local crossover, and socio-global crossover. A number of different procedures for these operations are available in the literature. The parameters and operations were selected in conformance with the values and operations described in Pradeepmon et al. (2016). The selected operations are explained below.

5.1. Mutation

The mutation operator modifies one or more values at randomly selected locations in randomly selected members of the population (swarm) with a probability, which normally is low, in agreement with its biological equivalent. The mutation operator updates the position of a particle based on its inertia thus moves in the direction of its own velocity. The mutation operators used in this paper is Swap Mutation (SWM) and is explained in detail.

The swap mutation operator randomly selects two random positions in the parent string and exchanges the elements in them (Banzhaf, 1990). As an example, let the parent solution string be (1 2 3 4 5 6), and suppose that the randomly selected positions are second and the fifth. The offspring resulting from swap mutation is (1 5 3 4 2 6).

```
SWM_pseudo_code
{
   1. Select parent for mutation.
   2. If rand() <= mutprob do
{
         a. Select two random positions c1 and c2 in parent.
         b. Exchange or swap the elements in positions c1 and c2 in parent to
            obtain offspring.
}
}
```

Fig. 3. Pseudo code of Swap Mutation

5.2. Crossover

The crossover operator is similar to natural reproduction which allows solutions to exchange information. Thus, the crossover operator receives two (or more) parent solutions and offspring solutions are produced from them. Crossover operator is applied to such that the position of the particle is updated to a better position by sharing knowledge with neighbours. The crossover operators incorporated in this paper are Position Based Crossover (POX), and Partially Mapped Crossover (PMX). A detailed discussion of each one of them is given below.

5.2.1. Position Based Crossover (POX)

The position based crossover selects a set of random positions from the parent chromosomes (Syswerda, 1991). In these random positions of one offspring, the genes of other parent are fixed. Further the missing elements are inserted in the same order as they appear in the parent. For example, consider the two parent chromosomes P1: (1 2 3 4 5 6) and P2: (3 5 1 6 2 4). The random positions selected are the first, third and the sixth positions. With this operator the following off-springs are generated: O1: (3 * 1 * * 4) and O2: (1 * 3 * * 6). Now, to complete O1 and O2, the missing elements are inserted in the same order as they appear in P1 and P2 respectively. This gives the offspring O1: (3 2 1 5 6 4) and O2: (1 5 3 2 4 6).

POX_pseudo_code

{

1. Select parent solutions $P1$ and $P2$ for crossover from current swarm $P(t)$.
2. Generate two empty strings $O1$ and $O2$ for offsprings.
3. Select a set of random positions, $R = (r1, r2,\ldots, rn)$, where n is also a random number less than the size of the parent.
4. Copy the elements in $P2$ occupying the positions represented by R to $O1$.
5. Insert the missing elements in $O1$ from $P1$ in the same order as they appear in $P1$.
6. Repeat steps 4 and 5 for $O2$ by exchanging the role of $P1$ and $P2$.

}

Fig. 4. Pseudo code of Position Based Crossover

5.2.2. Partially Mapped Crossover (PMX)

In Partially Mapped Crossover, the ordering information from parents is shared with offspring. It was proposed by Goldberg and Lingle (1985) in which a substring of one chromosome is mapped onto a corresponding substring of the other chromosome and the remaining information is exchanged. As an example, consider the following two chromosomes P1: (1 2 3 4 5 6) and P2 (2 4 6 5 3 1)

The PMX operator selects two random cut points along parents. Let the first cut point be selected between the first and the second elements, and the second cut point between the fourth and the fifth elements. For example, (1 | 2 3 4 | 5 6) and (2 | 4 6 5 | 3 1). The substrings between the cut points are then mapped to each other. In our example mappings are 2<->4, 3<->6, 4<->5. Now the mapping substrings are exchanged, i.e., the mapping substring of the first parent replaces that of the second offspring, and vice-versa. In our example, O1: (* | 4 6 5 | * *) and Offspring 2: (* | 2 3 4 | * *). Then, O1 is completed by copying the remaining elements of P1. If a facility is already present in the offspring it is replaced according to the mappings. For example, the first element of O1 would be a 1 like the first element of P1. But the fifth element 5 in P1 is already present in O1. Hence, because of the mapping 5<->4 and 4<->2 we choose the fifth element of O1 to be a 2. The sixth element of O1 would be a 6, which is already present. Because of the mappings 6<->3, it is chosen to be a 3. Hence, Offspring1: (1 | 4 6 5 | 2 3). In the same way, we get O2: (5 | 2 3 4 | 6 1).

```
PMX_pseudo_code

{
   1. Select parent solutions P1 and P2 for crossover from current swarm P(t).
   2. Select two cut points c1 and c2 randomly.
   3. Generate a mapping M of elements in P1 and P2, between the cut points c1 and c2.
   4. Copy the elements of P2 between c1 and c2 to O1.
   5. For i = 1 to sizeof(O1)
      {
         a. if element in position i of O1 is empty
            {
               i. if the element in position i of P1 is not present in O1
                  {
                     1. copy the element in position i of P1 to O1.
                  }
              ii. else
                  {
                     1. find the mapping of corresponding element from M and copy it to O1.
                  }
            }
         b. else
            {
               i. i = i + 1.
            }
      }
   6. Repeat steps 5 for O2.
}
```

Fig. 5. Pseudo code of Partially Mapped Crossover

5.3. Position Updating Strategy

Once, the possible position of particles based on inertia, cognition, local best and global best are found out, there are four different possible positions for the same particle. But, the particle can occupy only a single location and it is selected based on the position updating strategy. The strategy used in this paper is rank selection method. The rank selection process starts with ranking the swarm in hand and assigning fitness to each member in the swarm based on the rank of that member. The worst member of the swarm will get fitness value '1', the second worst will get fitness of '2' and so on, and the best will get a fitness value 'N' (where N is the swarm size). After ranking the swarm, all members have a selection probability proportional to their fitness value. But this method may lead to slower convergence of the algorithm, as the fitness of the better members does not differ much from other worse members. The operators and parameters for the proposed DPSO as selected from Pradeepmon et al. (2016), are given in Table 1.

Table 1
Selected operations and parameters for optimal solution

Sl. No.	Factor	Level Values
1	Position Updating Strategy (PUS)	Rank Selection (RKS),
2	Swarm Size (SMS)	2.5N (N is problem size)
3	Mutation Probability for Inertia (MPI)	0.9
4	Mutation Operator for Inertia (MOI)	Swap (SWM)
5	Crossover Probability for Cognition (CPC)	0.9
6	Crossover Operator for Cognition (COC)	Position Based (POX)
7	Crossover Probability for Socio-Local (CPL)	0.9
8	Crossover Operator for Socio-Local (COL)	Position Based (POX)
9	Crossover Probability for Socio-Global (CPG)	1
10	Crossover Operator for Socio-Global (COG)	Partially Mapped (PMX)
11	Termination Criterion (Maximum number of iterations)	100N

6. Illustration of the Proposed Algorithm

Fig. 6 represents the flow chart of the proposed DPSO algorithm. For a step by step illustration of the proposed algorithm, we consider a QAP with six facilities. This problem is derived from the benchmark problems on Dynamic Facility Layout problems provided by Balakrishnan and Cheng (2000). Table 2 gives the distance matrix of the departments and the material handling costs between departments for the illustration problem.

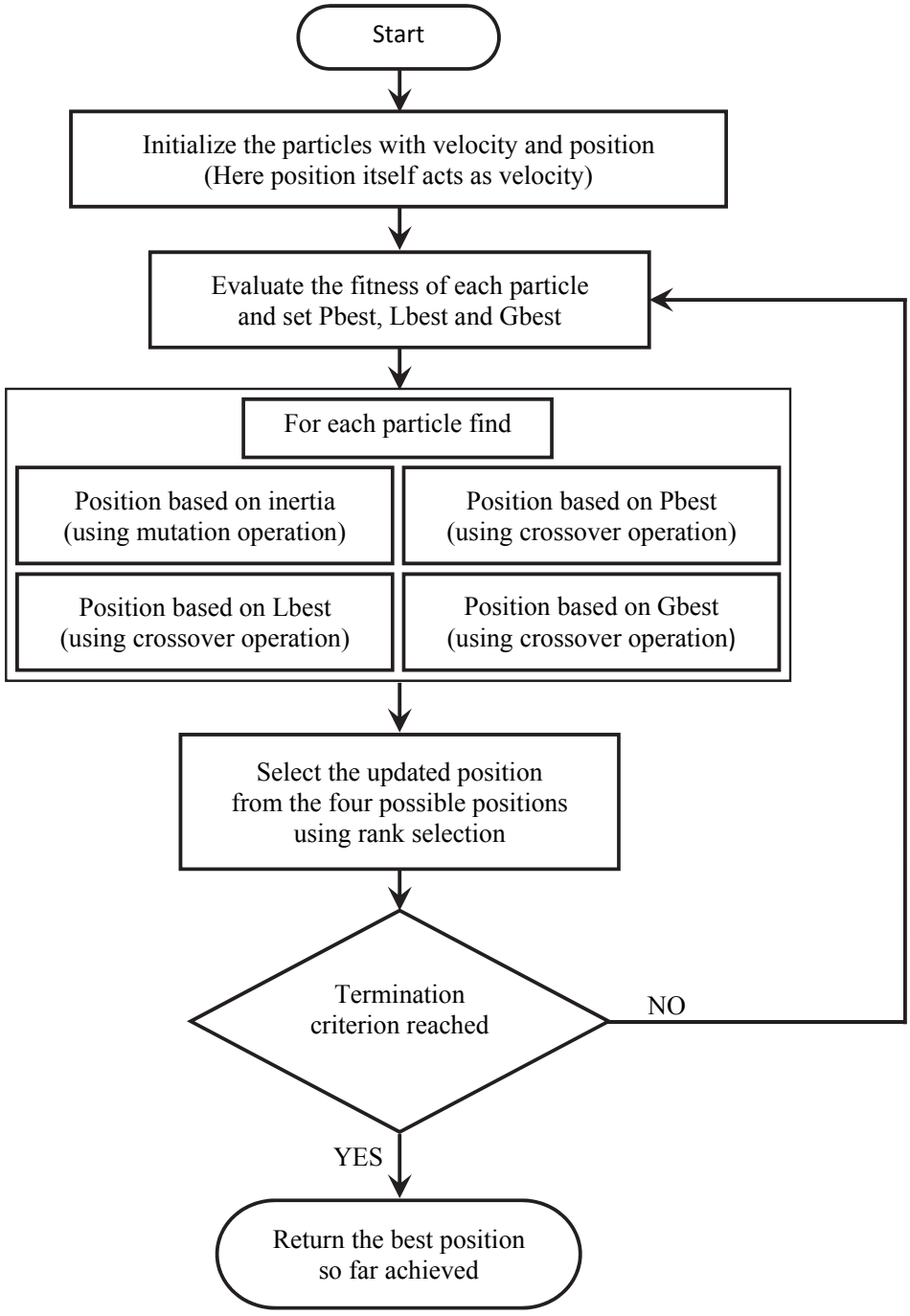

Fig. 6. Flowchart of proposed DPSO

Table 2
Data for illustration

Table 2(a) Distance matrix

From \ To	**1**	**2**	**3**	**4**	**5**	**6**
1	0	1	2	1	2	3
2	1	0	1	2	1	2
3	2	1	0	3	2	1
4	1	2	3	0	1	2
5	2	1	2	1	0	1
6	3	2	1	2	1	0

Table 2(b) Flow matrix

From \ To	**1**	**2**	**3**	**4**	**5**	**6**
1	0	90	689	194	165	494
2	668	0	1324	811	241	206
3	631	387	0	125	281	375
4	80	495	615	0	222	221
5	276	204	1127	490	0	676
6	109	409	1780	394	200	0

Step by step illustration of the algorithms is given below.

1. Initialise the parameters.

Swarm Size	=	2.5 N
Mutation Probability for Inertia	=	0.90
Crossover probability for personal best	=	0.90
Crossover probability for neighbourhood best	=	0.90
Crossover probability for global best	=	1.00
Maximum number of iterations	=	100 N
Number of Neighbourhoods	=	4

   These parameters are selected from Pradeepmon et al. (2016), in which the best combination of parameters and operations for the proposed DPSO were identified by using Taguchi's robust design methodology.

2. Generate an initial random swarm of 2.5N particles and calculate the objective function value as the material handling cost or transportation cost.

Particle	**Objective Function**
[6 2 4 1 5 3]	(24829)
[6 5 1 4 3 2]	(22298)
[6 2 4 1 5 3]	(24829)
[6 4 5 3 1 2]	(23702)
[3 2 5 1 6 4]	(23118)
[4 5 6 2 1 3]	(21745)
[3 4 1 5 2 6]	(25163)
[4 3 2 5 1 6]	(22876)
[3 6 4 1 2 5]	(23020)
[2 4 6 3 5 1]	(24075)
[3 6 1 4 5 2]	(23654)
[1 5 4 3 6 2]	(22031)
[4 6 5 2 3 1]	(20253)
[2 3 1 6 4 5]	(22852)
[3 2 6 5 4 1]	(23525)

3. Keep the current swarm as the personal best position (p_{best}) so far visited by each individual.
4. Divide the swarm into 4 groups.
5. Determine the best solution among the members (l_{best}) in each Neighbourhood.

Neighbourhood Best	OF
[6 5 1 4 3 2]	(22298)
[4 5 6 2 1 3]	(21745)
[1 5 4 3 6 2]	(22031)
[4 6 5 2 3 1]	(20253)

6. Determine the best solution (g_{best}) among the swarm.

   [4 6 5 2 3 1] 20253

7. Repeat the step 8 to step 10 till the termination criterion is satisfied
8. For each individual in the swarm find four possible movements by the following operations.
   a. Swap mutation of the individual with probability = 0.90

      Initial [6 2 4 1 5 3] (24829)

      After Mutation [6 1 4 2 5 3] (26893)

   b. Position based Crossover with p_{best} with probability = 0.90
      Better of the two offsprings saved
      Parents and Offsprings [6 2 4 1 5 3] (24829) (Since the personal best is the same as the particle)

   c. Position based Crossover with l_{best} with probability = 0.90

      Parents [6 2 4 1 5 3] (24829)
      [6 5 1 4 3 2] (22298)
      Offsprings [6 5 4 1 3 2] (20911)
      [6 2 5 1 4 3] (25163)
      Better of the two offsprings saved [6 5 4 1 3 2] (20911)

   d. Partially mapped Crossover with g_{best} with probability = 1.00

      Parents [6 2 4 1 5 3] (24829)
      [4 6 5 2 3 1] (20253)
      Offsprings [6 1 5 2 3 4] (22876)
      [3 6 4 1 5 2] (22785)
      Better of the two offsprings saved [3 6 4 1 5 2] (22785)

9. Select a movement among the four possible movements using the rank selection procedure and update the position of the individual as the resultant of the selected movement.
   [6 5 4 1 3 2] (20911)
10. Update (p_{best}), (g_{best}), and (l_{best}) using the updated swarm.
11. Once termination condition is reached, the best solution so far obtained is reported.
    [4 6 5 2 3 1] (20253)

7. Results and Discussions

The test instances taken from QAP library are solved using the DPSO. In this work, five sets of benchmark instances, namely, Bur, Had, Kra, Nug, and Rou which are available in QAP Library (Burkard et al., 1997) are considered. Each of the problems is solved ten times and the best, worst and average results obtained are reported with corresponding percentage deviation from the best-known solution. Table 3 presents the results of the experiments carried out.

Table 3
Results of the Computational Experiments

Sl. No.	Problem	Known Min	Solution Values			Percentage deviation from known min		
			Min	Max	Average	Min	Max	Average
1	bur26a	5426670	5434783	5450913	5441457	0.150	0.447	0.272
2	bur26b	3817852	3824420	3831477	3827786	0.172	0.357	0.260
3	bur26c	5426795	5428396	5453273	5436317	0.030	0.488	0.175
4	bur26d	3821225	3821419	3876647	3833711	0.005	1.450	0.327
5	bur26e	5386879	5387320	5435102	5402320	0.008	0.895	0.287
6	bur26f	3782044	3783123	3807692	3792180	0.029	0.678	0.268
7	bur26g	10117172	10118542	10168874	10145949	0.014	0.511	0.284
8	bur26h	7098658	7099677	7181427	7134108	0.014	1.166	0.499
9	had12	1652	1652	1676	1658.2	0.000	1.453	0.375
10	had14	2724	2724	2746	2729.2	0.000	0.808	0.191
11	had16	3720	3720	3740	3723.2	0.000	0.538	0.086
12	had18	5358	5362	5476	5398.6	0.075	2.202	0.758
13	had20	6922	6922	7066	6947.2	0.000	2.080	0.364
14	kra30a	88900	91160	97100	95002	2.542	9.224	6.864
15	kra30b	91420	93160	97460	95054	1.903	6.607	3.975
16	kra32	88700	91570	97340	94523	3.236	9.741	6.565
17	nug12	578	582	604	592.4	0.692	4.498	2.491
18	nug14	1014	1016	1072	1049.6	0.197	5.720	3.511
19	nug15	1150	1164	1190	1174.6	1.217	3.478	2.139
20	nug16a	1610	1630	1744	1677.6	1.242	8.323	4.199
21	nug16b	1240	1240	1338	1291.2	0.000	7.903	4.129
22	nug17	1732	1750	1812	1777.6	1.039	4.619	2.633
23	nug18	1930	1936	2032	1990.6	0.311	5.285	3.140
24	nug20	2570	2570	2718	2655.2	0.000	5.759	3.315
25	nug21	2438	2444	2594	2516.6	0.246	6.399	3.224
26	nug22	3596	3602	3788	3687.2	0.167	5.339	2.536
27	nug24	3488	3578	3740	3649	2.580	7.225	4.616
28	nug25	3744	3766	3920	3853.2	0.588	4.701	2.917
29	nug27	5234	5294	5492	5422.8	1.146	4.929	3.607
30	nug28	5166	5228	5522	5387	1.200	6.891	4.278
31	nug30	6124	6206	6434	6308.4	1.339	5.062	3.011
32	rou12	235528	240038	251196	245918.8	1.915	6.652	4.412
33	rou15	354210	364058	376222	371982.2	2.780	6.214	5.017
34	rou20	725522	738850	768966	756489.4	1.837	5.988	4.268

It is observed that in all the selected problems for computational study, the obtained best solution is closer to the known optimal solution with a maximum deviation of 3.24%. For most of the problems, the reported best solution is having a deviation of less than one percent from the best-known solution. Even in the case of maximum percentage deviation from the optimal solution the value is less than 10%.

The proposed algorithm is a variant of the simple DPSO and hence the results obtained are good when compared to other algorithms. Out of the 34 problems considered for only four problems the best solution varies from the best-known solution by a factor of more than 2.0%. For all other problems, the percentage variation is less than 2.0% with 21 problems reporting less than 1.0% variation from the best-known solution. For eight problems even the worst solution shows a variation of less than 1.0% from the best-known solution.

8. Conclusions

Quadratic assignment problem is one of the most complex combinatorial optimization problems. Many metaheuristic algorithms have been used for solving QAPs. The metaheuristics perform better when hybridised than when they stand alone. But, this work is an improvement of general DPSO without using any hybridization and in this work, a modified discrete particle swarm optimization algorithm is proposed for solving QAPs. The proposed algorithm varies from the original DPSO in terms of the position updating strategy. In classical DPSO only one updated position is found out incorporating all components

of information available with the particle. But in the proposed method four different possible positions are found out by using the four data stored in the particle, one at a time. Out of these four possible positions, one is selected according to the fitness of the solution represented by the new positions.

The DPSO thus obtained by incorporating the modified position updating procedure is then employed for solving five sets of QAPs, namely, Bur, Had, Kra, Nug, and Rou, available in QAP library. The results are listed and the variations are minute with most of the problems giving less than one percentage deviation from the best-known solution. This paper works with a simple DPSO and an intensification strategy in the search procedure may increase the efficiency of the algorithm. Further researches, employing DPSO for hybridization with other methods, can be carried out using the reported combination of parameters and operations for better results.

References

Ahmed, Z. H. (2015a). A multi-parent genetic algorithm for the quadratic assignment problem. *OPSEARCH*, *52*(4), 714–732.

Ahmed, Z. H. (2015b). An improved genetic algorithm using adaptive mutation operator for the quadratic assignment problem. *38th International Conference on Telecommunications and Signal Processing (TSP), 2015*, 1–5. IEEE.

Anderson, E. J. (1996). Mechanisms for local search. *European Journal of Operational Research*, *88*(1), 139–151.

Arkin, E. M., Hassin, R., & Sviridenko, M. (2001). Approximating the maximum quadratic assignment problem. *Information Processing Letters*, *77*(1), 13–16.

Asarry, A., Zain, M. Z. M., Mailah, M., & Hussein, M. (2013). Suppression of hand tremor model using active force control with Particle swarm optimization and differential evolution. *International Journal of Innovative Computiong, Information and Control*, *9*(9), 3759–3777.

Azarbonyad, H., & Babazadeh, R. (2014). A Genetic Algorithm for solving Quadratic Assignment Problem(QAP). *Computing Research Repository*, *abs/1405.5050*.

Balakrishnan, J., & Cheng, C. H. (2000). Genetic search and the dynamic layout problem. *Computers and Operations Research*, *27*(6), 587–593.

Banzhaf, W. (1990). The "molecular" traveling salesman. *Biological Cybernetics*, *64*(1), 7–14.

Battiti, R., & Tecchiolli, G. (1994). Simulated annealing and tabu search in the long run: A comparison on QAP tasks. *Computer and Mathematics with Applications*, *28*(6), 1–8.

Bazaraa, M. S., & Sherali, H. D. (1982). On the Use of Exact and Heuristic Cutting Plane Methods for the Quadratic Assignment Problem. *The Journal of the Operational Research Society*, *33*(11), 991–1003.

Brixius, N. W., & Anstreicher, K. M. (2000). Solving Quadratic Assignment Problems Using Convex Quadratic Programming Relaxations. *Optimization Methods and Software*, *16*, 49–68.

Burkard, R. E., Dell'Amico, M., & Martello, S. (2009). *Assignment Problems*. Philadelphia: Society for Industrial and Applied Mathematics.

Burkard, R. E., Karisch, S. E., & Rendl, F. (1997). QAPLIB – A Quadratic Assignment Problem Library. *Journal of Global Optimization*, *10*(4), 391–403.

Burkard, R. E., & Rendl, F. (1984). A thermodynamically motivated simulation procedure for combinatorial optimization problems. *European Journal of Operational Research*, *17*(2), 169–174.

Clausen, J., & Perregaard, M. (1997). Solving Large Quadratic Assignment Problems in Parallel. *Computational Optimization and Applications*, *8*(2), 111–127.

Connolly, D. T. (1990). An improved annealing scheme for the QAP. *European Journal of Operational Research*, *46*(1), 93–100.

Consoli, S., Moreno-Pérez, J. A., Darby-Dowman, K., & Mladenovic, N. (2010). Discrete Particle Swarm Optimization for the minimum labelling Steiner tree problem. *Natural Computing*, *9*(1), 29–46.

Czapiński, M. (2013). An effective Parallel Multistart Tabu Search for Quadratic Assignment Problem on CUDA platform. *Journal of Parallel and Distributed Computing*, *73*(11), 1461–1468.

Day, R. O., Kleeman, M. P., & Lamont, G. B. (2003). Solving the multi-objective quadratic assignment problem using a fast messy genetic algorithm. *Proceedings of Congress Evolutionary Computation (CEC '03)*, *4*, 2277–2283.

De, A., Bhattacharjee, A. K., Chanda, C. K., & Maji, B. (2012). Hybrid particle swarm optimization with wavelet mutation based segmentation and progressive transmission technique for MRI images. *International Journal of Innovative Computing, Information and Control*, *8*(7), 5179–5197.

Drezner, Z. (2005). The extended concentric tabu for the quadratic assignment problem. *European Journal of Operational Research*, *160*(2), 416–422.

Drezner, Z. (2008). Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem. *Computers & Operations Research*, *35*(3), 717–736.

Drezner, Z., & MisevicIus, A. (2013). Enhancing the performance of hybrid genetic algorithms by differential improvement. *Computers & Operations Research*, *40*(4), 1038–1046.

Eberhart, R. C., & Shi, Y. (2001). Particle swarm optimization: developments, applications and resources. *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, *1*, 81–86. IEEE.

Fleurent, C., & Glover, F. (1999). Improved Constructive Multistart Strategies for the Quadratic Assignment Problem Using Adaptive Memory. *INFORMS Journal on Computing*, *11*(2), 198–204.

Gambardella, L. M., Taillard, É. D., & Dorigo, M. (1999). Ant Colonies for the Quadratic Assignment Problem. *The Journal of the Operational Research Society*, *50*(2), 167–176.

Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co.

Goldberg, D. E., & Lingle Jr., R. (1985). Alleles, loci, and the traveling salesman problem. In J. J. Grefenstette (Ed.), *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. Lawrence Erlbaum Associates, Publishers.

Hafiz, F., & Abdennour, A. (2016). Particle Swarm Algorithm variants for the Quadratic Assignment Problems-A probabilistic learning approach. *Expert Systems with Applications*, *44*, 413–431.

Hahn, P. M., Hightower, W. L., Johnson, T. A., Guignard-Spielberg, M., & Roucairol, C. (2001). Tree elaboration strategies in branch and bound algorithms for solving the quadratic assignment problem. *Yugoslav Journal of Operational Research*, *11*(1), 41–60.

Hassin, R., Levin, A., & Sviridenko, M. (2009). Approximating the minimum quadratic assignment problems. *ACM Transactions on Algorithms*, *6*(1), 18:1–18:10.

Hong, G. (2013). A Hybrid Ant Colony Algorithm for Quadratic Assignment Problem. *The Open Electrical and Electronic Engineering Journal*, *7*, 51–54.

Izakian, H., Ladani, B. T., Abraham, A., & Snášel, V. (2010). A Discrete Particle Swarm Optimization Approach for Grid Job Scheduling. *International Journal of Innovative Computing, Information and Control*, *6*(9), 1–15.

James, T., Rego, C., & Glover, F. (2009). A cooperative parallel tabu search algorithm for the quadratic assignment problem. *European Journal of Operational Research*, *195*, 810–826.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. *Proceedings of the IEEE International Conference on Neural Networks*, 1942–1948.

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

Li, & Smith, J. M. (1995). Theory and Methodology: An algorithm for Quadratic Assignment Problems. *European Journal of Operational Research*, *81*, 205–216.

Lian, Z., Lin, W., Gao, Y., & Jiao, B. (2014). A Discrete Particle Swarm Optimization Algorithm for Job-shop Scheduling Problem to Maximizing Production. *International Journal of Innovative Computing, Information and Control*, *10*(2), 729–740.

Liu, H., & Abraham, A. (2007). A Hybrid Fuzzy Variable Neighborhood Particle Swarm Optimization Algorithm. *Journal of Universal Computer Science*, *13*.

Loiola, E. M., Abreu, N. M. M. de, Boaventura-Netto, P. O., Hahn, P., & Querido, T. (2007). A survey for the quadratic assignment problem. *European Journal of Operational Research*, *176*(2), 657–690.

Loiola, E. M., Maria, N., Abreu, M., Boaventura-netto, P. O., Hahn, P., & Querido, T. (2004). *An Analytical Survey for the Quadratic Assignment Problem*. Council for the Scientific and Technological Development, of the Brazilian Gov.

Mamaghani, A. S., & Meybodi, M. R. (2012). Solving the Quadratic Assignment Problem with the modified hybrid PSO algorithm. *Proceedings of 6th International Conference on Application of Information and Communication Technologies (AICT)*, 1–6.

Matsui, S., Kobayashi, Y., Watanabe, K., & Horio, Y. (2004). Exponential chaotic tabu search hardware for quadratic assignment problems using switched-current chaotic neuron IC. *Proceedings of IEEE Internationa Joint Conference on Neural Networks*, *3*, 2221–2225.

Misevicius, A. (2000). An Intensive Search Algorithm for the Quadratic Assignment Problem. *Informatica*, *11*(2), 145–162.

Misevicius, A. (2003). A Modified Simulated Annealing Algorithm for the Quadratic Assignment Problem. *Informatica*, *14*(4), 497–514.

Misevicius, A. (2004). An improved hybrid optimization algorithm for the quadratic assignment problem. *Mathematical Modelling and Analysis*, *9*(2), 149–168.

Misevicius, A., & Guogis, E. (2012). Computational study of four genetic algorithm variants for solving the quadratic assignment problem. *International Conference on Information and Software Technologies*, 24–37. Springer.

Osman, I. H., & Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research*, *63*(5), 511–623.

Ozsoydan, F. B., & Sarac, T. (2011). A Discrete Particle Swarm Optimization Algorithm for Bi-Criteria Warehouse Location Problem. *Istanbul University Econometrics and Statistics e-Journal*, *13*(1), 114–124.

Pan, Q.-K., Tasgetiren, M. F., & Liang, Y.-C. (2008). A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers and Operations Research*, *35*(9), 2807–2839.

Paul, G. (2011). An efficient implementation of the simulated annealing heuristic for the quadratic assignment problem. *Computing Research Repository*, *abs/1111.1353*.

Paul, G. (2012). A GPU implementation of the Simulated Annealing Heuristic for the Quadratic Assignment Problem. *Computing Research Repository*, *abs/1208.2675*.

Peng, H., Zhang, Z., Wang, J., & Shi, P. (2013). Audio watermarking framework using multi-objective particle swarm optimization. *International Journal of Innovative Computing, Information and Control*, *9*(7), 2789–2800.

Peng, T., Huanchen, W., & Dongme, Z. (1996). Simulated Annealing for the Quadratic Assignment Problem: A further study. *Computers and industrial Engineering*, *31*(3/4), 925–928.

Pradeepmon, T. G., Panicker, V. V., & Sridharan, R. (2016). Parameter Selection of Discrete Particle Swarm Optimization Algorithm for the Quadratic Assignment Problems. *Procedia Technology*, *25*, 998–1005.

Ramkumar, A. S., Ponnambalam, S. G., Jawahar, N., & Suresh, R. K. (2008). Iterated fast local search algorithm for solving quadratic assignment problems. *Robotics and Computer-Integrated Manufacturing*, *24*(3), 392–401.

Rapai, M. R., Kanovi, Ž., & Jelici, Z. D. (2008). Discrete particle swarm optimization algorithm for solving optimal sensor deployment problem. *Journal of Automatic Control*, *18*(1), 9–14.

See, P. C., & Wong, K. Y. (2008). Application of ant colony optimisation algorithms in solving facility layout problems formulated as quadratic assignment problems: a review. *International Journal of Industrial and Systems Engineering*, *3*(6), 644–672.

Sevkli, F. Mehmet Mamedsaidov Ruslan Camci. (2014). A novel discrete particle swarm optimization for p-median problem. *Journal of King Saud University - Engineering Sciences*, *26*(1), 11–19.

Skorin-Kapov, J. (1990). Tabu search applied to the quadratic assignment problem. *ORSA Journal on Computing*, *2*(1), 33–45.

Song, L. Q., Lim, M. H., & Ong, Y. S. (2011). Neural meta-memes framework for managing search algorithms in combinatorial optimization. *IEEE Workshop on Memetic Computing (MC), 2011*, 1–6.

Syswerda, G. (1991). Schedule Optimization Using Genetic Algorithms. In L. Davis (Ed.), *Handbook of Genetic Algorithms*. New York, NY: Van Nostrand Reinhold.

Szwed, P., & Chmiel, W. (2015). Multi-swarm PSO algorithm for the Quadratic Assignment Problem: a massive parallel implementation on the OpenCL platform. *arXiv preprint arXiv:1504.05158*.

Szwed, P., Chmiel, W., & Kadłuczka, P. (2015). OpenCL Implementation of PSO Algorithm for the Quadratic Assignment Problem. *Artificial Intelligence and Soft Computing*, 223–234. Springer.

Talbi, E. G., Hafidi, Z., & Geib, J.-M. (1998). A parallel adaptive tabu search approach. *Parallel Computing*, *24*(14), 2003–2019.

Tasgetiren, M. F., Liang, Y.-C., Sevkli, M., & Gencyilmaz, G. (2006). Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem. *International Journal of Production Research*, *44*(22), 4737–4754.

Tate, D. M., & Smith, A. E. (1995). A genetic approach to the quadratic assignment problem. *Computers and Operations Research*, *22*(1), 73–83.

Tosun, U. (2014). A New Recombination Operator for the Genetic Algorithm Solution of the Quadratic Assignment Problem. *Procedia Computer Science*, *32*(0), 29–36.

Tseng, & Liang, S. C. (2006). A Hybrid Metaheuristic for the Quadratic Assignment Problem. *Computational Optimization and Applications*, *34*, 85–113.

Urban, T. L. (1998). Solution procedures for the dynamic facility layout problem. *Annals of Operations Research*, *76*(0), 323–342.

Uwate, Y., Nishio, Y., Ueta, T., Kawabe, T., & Ikeguchi, T. (2004). Performance of Chaos and Burst Noises Injected to the Hopfield NN for Quadratic Assignment Problems. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *E87-A*(4), 937–943.

Wakabayashi, S., Kimura, Y., & Nagayama, S. (2006). FPGA implementation of tabu search for the quadratic assignment problem. *Proceedings of IEEE International Conference on Field Programmable Technology (FPT 2006)*, 269–272.

West, D. H. (1983). Algorithm 608: Approximate Solution of the Quadratic Assignment Problem. *ACM Transactions on Mathematical Software*, *9*(4), 461–466.

Wilhelm, M. R., & Ward, T. L. (1987). Solving Quadratic Assignment Problems by 'Simulated Annealing. *IIE Transactions*, *19*(1), 107–119.

Wu, Y., & Ji, P. (2007). Solving the quadratic assignment problems by a genetic algorithm with a new replacement strategy. *International Journal of Humanities and Social Science*, 151–155.

Xia, Y. (2010). An efficient continuation method for quadratic assignment problems. *Computers and Operations Research*, *37*(6), 1027–1032.

Zaied, A. N. H., & Shawky, L. A. E.-F. (2014). A Survey of the Quadratic Assignment Problem. *International Journal of Computer Applications*, *101*(6), 28–36.

Zhang, Y., Wang, S., & Ji, G. (2015). A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. *Mathematical Problems in Engineering*, 1–38.

Zhu, W., Curry, J., & Marquez, A. (2010). SIMD tabu search for the quadratic assignment problem with graphics hardware acceleration. *International Journal of Production Research*, *48*(4), 1035–1047.