PARTICLE SWARM OPTIMIZATION COMBINED WITH LOCAL SEARCH AND VELOCITY RE-INITIALIZATION FOR SHORTEST PATH COMPUTATION IN NETWORKS

Ammar W. Mohemmed Nirod Chandra Sahoo
Faculty of Engineering and Technology
Multimedia University, 75450 Melaka, Malaysia
E-mail: ammar.wmohemmed@mmu.edu.my ; nirodchandra.sahoo@mmu.edu.my

Abstract - This paper presents the application of particle swarm optimization (PSO) based search algorithm for solving the single source shortest path problem (SPP) commonly encountered in graph theory. A new particle encoding/decoding scheme has been devised for representing the SPP parameters as a particle. In order to enhance the search capability of PSO, a selective local search mechanism and periodic velocity re-initialization of particles have been incorporated. Simulation results on several networks with random topologies are used to illustrate the efficiency of the proposed hybrid PSO algorithm for computation of shortest paths in networks.

I.INTRODUCTION

Shortest path computation is one of the most fundamental problems in graph theory. The huge interest in the problem is mainly due to the wide spectrum of its applications, ranging from routing in communication networks to robot motion planning, scheduling, sequence alignment in molecular biology and length-limited Huffman coding, to name only a very few. Deo and Pang [1] have surveyed a large number of algorithms and applications of the shortest path problems.

The shortest path problem is defined as follows. An undirected graph $G = (V, E)$ comprises a set of nodes $V = \{v_i\}$ and a set of edges $E \in V \times V$ connecting nodes in V. Corresponding to each edge, there is a nonnegative number C_{ij} representing the cost (distance, transit times, etc) of the edge from node v_i to node v_j. A path from node v_i to node v_k is a sequence of nodes $(v_i, v_j, v_l, \cdots, v_k)$ from E in which no node appears more than once. For example, in Fig. 1, a path from node 1 to node 7 is represented as (1, 4, 3, 7). The shortest path problem is to find a path between two given nodes having minimum total cost. Let 1 denote the initial (starting) node and n denote the end node of the path. The integer programming model of the shortest path problem (SPP) is formulated as follows:

$$\min \sum_{(i,j)\in E} C_{ij} x_{ij}$$

such that

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \begin{cases} 1, i = s \\ -1, i = t \\ 0, i \neq s, t \end{cases} \quad (1)$$

where x_{ij} is 1 if the edge connecting nodes i and j is in the path or 0 otherwise; s and t stand for source and terminal node, respectively.

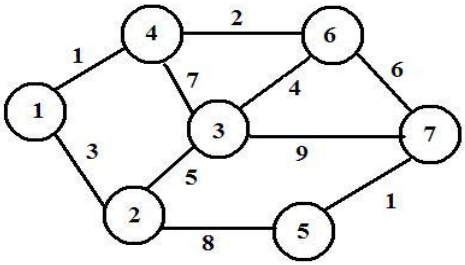

Fig .1: A network with 7 nodes and 10 edges

Deterministic algorithms such as Dijkstra and Bellman-Ford algorithms [2] are mostly used to solve the single source shortest path problem. With the developments of communication, computer science, transportation systems, and so on, more variants of shortest path based problems have appeared. Some of these problems include traveling salesman problem, K-shortest paths, constrained shortest path problems, multiple objective shortest path problem, and network flow problems etc. Most of these problems are NP-hard. Therefore, polynomial-time algorithms for these problems are impossible. Thus, the evolutionary and heuristics algorithms are the most attractive alternative ways to go for.

Among the notable heuristic algorithms for path finding optimization problems in network graphs, successful use of genetic algorithm (GA) and Tabu Search (TS) has been reported [3–7]. Recently, another powerful heuristic optimization algorithm, called particle swarm optimization [8], has been used for many optimization problems with high success. Some comparative studies of the performances of GA and PSO have also been reported [9 -12]. All these studies have firmly established the same

1-4244-0708-7/07/$20.00 ©2007 IEEE

effectiveness of PSO compared to GA. In fact, for some problems, PSO performance is reported to be superior. It is also to be noted that GA and TS demand expensive computational cost. In contrast, PSO requires less computational bookkeeping and generally only a few lines of implementation code.

To best knowledge of the authors, there is no such report published on the use of PSO for the shortest path problem. It is the purpose of this paper to investigate this. In this work, a PSO-based hybrid search algorithm has been designed for computation of single source shortest paths in networks. A new particle position representation and decoding scheme have been formulated for this specific problem. Further, in order to speed up the search process, a selective local search and periodic velocity re-initialization strategy have been incorporated into the main PSO. The effectiveness of the proposed hybrid PSO algorithm for finding shortest paths in networks is verified by computer simulation studies on different random network topologies.

II. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization is a population based stochastic optimization technique inspired by the social behavior of bird flock (and fish school etc.), as developed by Kennedy and Eberhart in 1995 [8].

The algorithmic flow in PSO starts with a population of particles whose positions, that represent the potential solutions for the studied problem, and velocities are randomly initialized in the search space. The search for optimal position (solution) is performed by updating the particle velocities, hence positions, in each iteration according to the following two equations:

$$PV_{id} = PV_{id} + \varphi_1 r_1 (B_{id} - X_{id}) + \varphi_2 r_2 (B^n_{id} - X_{id});$$

$$i = 1, 2, \cdots, N_s \text{ and } d = 1, 2, \cdots, D \quad (2)$$

$$X_{id} = X_{id} + PV_{id} \quad (3)$$

where φ_1 and φ_2 are positive constants, called *acceleration coefficients*, N_s is the total number of particles in the swarm, D is the dimension of problem search space, i.e., number of parameters of the function being optimized, r_1 and r_2 are two independently generated random numbers in the range [0, 1] and "n" represents the index of the best particle in the neighborhood of a particle. The other vectors are defined as: $\mathbf{X}_i = [X_{i1}, X_{i2}, \cdots, X_{iD}]$ ≡ Position of i-th particle; $\mathbf{PV}_i = [PV_{i1}, PV_{i2}, \cdots, PV_{iD}]$ ≡ Velocity of the i-th particle; $\mathbf{B}_i = [B_{i1}, B_{i2}, \cdots, B_{iD}]$ ≡ Best position of the i-th particle ($pBest_i$), and $\mathbf{B}^n_i = [B^n_{i1}, B^n_{i2}, \cdots, B^n_{iD}]$ ≡ Best position found by the neighborhood of the particle i ($nBest_i$). The pseudo-codes for general algorithmic flow of PSO are listed in Fig. 2.

```
Initialize the position and velocity randomly.
Calculate fitness value of each particle.
Calculate pBest and nBest for each particle.
While(iteration_count < max_iteration)
  For each particle,
     Update particle velocity using Eq. (2).
     Update particle position using Eq. (3).
     Calculate fitness value of particle.
     Update pBest if its current fitness value is better
     than its pBest.
     Update nBest, i.e., choose the position of the
     particle with the best fitness value among all its
     neighbors as the nBest for a specific neighborhood
     topology.
End while
```

Fig. 2: Pseudo-codes for general algorithmic flow of Particle Swarm Optimization

Eq. (2) calculates a new velocity for each particle based on its previous velocity, the particle's position at which the best possible fitness has been achieved so far, and the neighbors' best position achieved. Eq. (3) updates each particle's position in the solution hyperspace. φ_1 and φ_2 are two learning factors, which control the influence of *pBest* and *nBest* on the search process. In all initial studies of PSO, both φ_1 and φ_2 are taken to be 2.0. However, in most cases, the velocities quickly attain very large values, especially for particles far from their global best. As a result, particles have larger position updates with particles leaving boundary of the search space. To control the increase in velocity, velocity clamping is used in Eq. (2). Thus, if the right side of Eq. (2) exceeds a specified maximum value PV_d^{max}, then the velocity on that dimension is clamped to PV_d^{max}. Many improvements have been incorporated into this basic algorithm. In [13], Maurice proposed the use of a constriction factor χ; the algorithm was named the constriction factor method (CFM) where Eq. (2) is modified as:

$$PV_{id} = \chi[PV_{id} + \varphi_1 r_1 (B_{id} - X_{id}) + \varphi_2 r_2 (B^n_{id} - X_{id})] \quad (4)$$

where $\chi = 2\left(\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|\right)^{-1}$ if $\varphi = \varphi_1 + \varphi_2 > 4$ (5)

The objective behind the use of constriction factor is to prevent the velocity from growing out of bounds, thus the velocity clamping is not required. But, Eberhart and Shi [14] have reported that the best performance can be

achieved with constriction factor and velocity clamping. The PSO's implementation is very simple. To pose a problem in PSO framework, the important step is to devise a coding scheme for particle representation, which is discussed next for the SPP.

III. PSO-BASED ALGORITHM FOR SHORTEST PATH COMPUTATION

In this section, the proposed PSO-based hybrid algorithm using selective local search and periodic velocity re-initialization is described. The basic module of PSO is as described in the previous section. The main components of the proposed algorithm are: particle representation, selective local search and periodic velocity re-initialization; these are discussed in detail.

A. Cost-Biased Particle Encoding/Decoding

There is a limited literature for using PSO to solve combinatorial optimization problems [15-16]; furthermore, the research results on computing shortest path by use of PSO is not known till date. Therefore, research works using genetic algorithm (GA) are referred here. One crucial issue in using GA to solve combinatorial problems is chromosome representation of the problem as it is an important step for success of the algorithm. Chromosome encoding can be categorized into direct and indirect encoding. Direct encoding means that a chromosome of the GA contains a gene for each item of the problem represented by item ID (identification number). In this case, the chromosome length is variable. This type of encoding was used in [3]. Thus, for Fig. 1, a chromosome that represents a solution (path) from node 1 to node 7 may have gene values (1, 4, 6, 7) or (1, 2, 3, 6, 7) etc.

On the other hand, in indirect encoding, the chromosome encodes guiding information about the solution rather than the solution itself. One advantage of this encoding is that the chromosome size is fixed, but the method needs a *decoding* procedure to extract the real solution from the guiding information. Two variants of this indirect encoding exist in literature: *Priority* encoding and *Weight* encoding. In [4], a priority encoding is proposed for a genetic SPP algorithm. In this encoding, the position of the gene in the chromosome represents the node ID, while the value of the gene is a number representing the ***priority*** of the node. The path is constructed in *a path growth* procedure by appending nodes starting from the source. At each step, the next node with the higher priority is chosen from those which have direct links with the current node. The procedure continues until the destination node is reached. The best chromosome at the end of a run of the algorithm is that one which contains priorities that lead the decoding procedure to select nodes forming the shortest path. One disadvantage of this encoding is that the chromosome is "indirectly" encoded; it does not include information about the graph or the network's characteristics like its edges' costs. Another variant of indirect coding of the chromosome is called ***weighted encoding*** [5]. Similar to the priority encoding, the chromosome is a vector of values called *weights*. This vector is used to modify the problem parameters, for instance the cost of the edges. First, the original problem is temporarily modified by biasing the problem parameters with the weights. Secondly, a problem-specific non-evolutionary *decoding heuristic* is used to actually generate a solution for the modified problem. This solution is finally interpreted and evaluated for the original (unmodified) problem.

Inspired by the above two encoding schemes, a representation scheme, referred to as *Cost-Biased* encoding/decoding, is devised to suit the swarm particles for SPP. Note that direct encoding is not appropriate for the swarm particles, as the updating for the particles are based on arithmetic operations in Eqs. (3) and (4). The encoding of the particle is based on bias values of the nodes and the decoding of the particle is based on the path growth procedure which takes into account the bias values of the nodes as well as cost of the edges. The particle contains a vector of node bias values; hence length of the particle equals to the number of nodes. To construct a path from an arbitrary particle, starting from the initial node (node 1) to the final node (node n), edges are appended into the path consecutively. At each step, the next node (node j) is selected from the nodes having direct links with the current node such that the product of the (next) node bias (ρ_j) and the edge cost is minimum, i. e.,

$$j = \min\left\{C_{ij}\rho_j \;/ (i,j) \in E\right\},\ \rho_j \in [-1.0, 1.0] \tag{6}$$

The steps of this algorithm are summarized in Fig. 3. Note that the bias values can take negative or positive real numbers in the range [-1.0, 1.0]. It is seen that the problem parameters (link costs) are part of the encoding/decoding procedure. Unlike the priority encoding where a node is appended to the partial path based only on its priority. But, in the proposed procedure, a node is appended to the path based on the minimum of the product of the node (next node) bias and the link cost that connects the current node with the next one to be selected. Experimental results show superiority of this procedure over the priority encoding when it is implemented within PSO frame. The PSO-based search is performed for optimal set of node bias values that result in shortest path in a given network.

```
Particle_Decoding (Solution)
Let i = 1, k =1, P(k) = {1} // P denotes the path
While (i ≠ n ) {
    k = k + 1
    j = the node has direct link with node i && has the
     minimum (C_{i,j} ρ_j ) value
    i = j, {P(k), {i}} → P(k) } }
end while
```

Fig. 3: Pseudo-codes for cost-biased encoding/decoding procedure

B. Local Search

Evolutionary algorithms (EA) are robust and powerful global optimization techniques for solving large-scale problems that have many local optima. However, they require high CPU times and are mostly poor in terms of convergence performance. On the other hand, local search algorithms can converge in a few iterations but lack a global perspective. The combination of global and local search procedures should offer the advantages of both optimization methods while offsetting their disadvantages [17]. The hybridizing of evolutionary algorithms with local search operators that work within the EA loop has been termed "Memetic Algorithms". Memetic Algorithms have been shown to be faster and more accurate than EAs on some problems. Greedy Randomized Adaptive Search Procedure (GRASP) [18] is a multi-start metaheuristic for combinatorial problems in which each iteration consists basically of two phases: construction and local search. The construction phase builds a feasible solution whose neighborhood is investigated until a local minimum is found during the local search phase. Therefore, incorporating a local search within the main algorithm proves to be beneficial in obtaining better results.

A sort of local search is incorporated in the proposed PSO-based algorithm for the SPP. The local search is called in every iteration by each particle when that particle has a better solution than that obtained in the previous iteration. The local search is based on edge elimination and uses a binary activation matrix of size (N x N), where N is the number of nodes in the network. When the binary bit in the activation matrix at position (i, j) is set to zero, that edge will not be included in the decoding procedure. The edges of the considered path are chosen for elimination one at a time in sequence by setting the corresponding activation bit to zero and then the Particle_Decoding procedure is used construct a new path. If the new path constructed is shorter than the original path, it will be recorded and the fitness of the particle will be modified to the fitness of this new locally obtained path and all the relevant information for obtaining this new path (the bias vector and the edge that has been banned from consideration during local search path construction) are stored. The pseudocodes for local search process are given in Fig. 4.

```
Local_Search(Solution)
 for k = 1,….., No_of_Nodes_in_path Do
     Set edge (k, k+1) as non-active.
     Construct a new path using the particle bias vector
     and Particle_Decoding procedure.
     If the new path has better fitness
         {Update the particle fitness with the fitness of
         new path;
         Store all relevant information (bias vector and
         banned edge);}
     Set edge (k, k+1) active
 end for;
```

Fig. 4: Pseudocodes for local search procedure

```
Generate initial population of particles.
Calculate fitness value of each particle.
Calculate pBest and nBest for each particle.
While(iteration_count < max_iteration)
    Update velocity of each particle using Eq. (4).
    Update position of each particle using Eq. (3).
    Calculate fitness value of each particle.
        {Call Particle_Decoding(particle)
          Calculate particle's fitness.
        }
    Update pBest for each particle if its current fitness
    value is better than its pBest;
    if it is better, call Local_Search(particle)
    Update nBest for each particle, i.e., choose the
    position of the particle with the best fitness value
    among all its neighbors as the nBest for a specific
    neighborhood topology.
    if (iteration_count mod 10) ≡ 0, then re-initialize
      velocities of all particles.
end while
```

Fig. 5: Pseudocodes of PSO algorithm combined with local search and velocity re-initialization

C. Velocity Re-initialization

One of the problems of the PSO is the premature convergence to a local minimum. It does not continue to improve on the quality of solutions after a certain number of iterations have passed [19]. As a result, the swarm becomes stagnated after a certain number of iterations and may end up with a solution far from optimality. Gregarious PSO [20] avoids premature convergence of the swarm; the particles are re-initialized with a random velocity when stuck at a local minimum. Dissipative PSO [21] reinitializes the particle positions at each iteration with a small probability (0.001). In [22], this additional perturbation is carried out with different probabilities

based on time-dependent strategy. In present study, the velocities of all the particles (in all the dimensions) are reinitialized randomly to the range $\left(-PV_d^{max}, PV_d^{max}\right)$ periodically after a certain number of iterations (say, 10 iterations). Effects of changing the iteration number at which the velocity re-initialization is done and auto-tuning of this parameter are under study. The pseudocodes of the PSO combined with velocity re-initialization and local search are listed in Fig. 5. The fitness of each particle is computed as:

Fitness = (Length/cost of the path represented by the particle)$^{-1}$ = (sum of the cost of all the edges in the path)$^{-1}$

IV. EXPERIMENTAL RESULTS

To evaluate the performance of the proposed hybrid PSO algorithm for SPP, different network topologies are generated using Waxman model [23] for random topology generator. For each network with certain number of nodes and edges, 20 random topologies are generated with different seed numbers. Edge costs are taken randomly from the range [1, 1000]. The other PSO parameters are chosen as: Number of Particles = 100; Neighborhood Topology = Ring; φ_1 equals 2.0 and φ_2 is chosen to be 2.2; thus φ = 4.2. From Eq. (5), χ = 0.74; Maximum number of iterations = 500; maximum velocity = ±1.0. The performance of the algorithm is assessed by success rate which is defined as the average number of times over 20 runs the shortest path is found. The shortest path is also computed separately using Dijkstra algorithm for comparison purposes. Also, the average number of iterations to get the shortest path is recorded for each network topology as this offers a rough estimate of the CPU time required. The results are compared for four different cases of algorithm operation as follows: one case of traditional GA (reported for this problem in [4]) and three different implementations of PSO

Case 1: Priority-based encoding is used in GA [4].
Case 2: Priority-based encoding is used in standard PSO.
Case 3: Proposed cost-biased particle encoding/decoding is used in standard PSO.
Case 4: Proposed cost-biased particle encoding/decoding is used in standard PSO combined with selective local search and periodic velocity re-initialization.

The results compared in Table 1 show the superiority of the proposed encoding/decoding scheme over the priority encoding whether implemented with PSO frame or GA frame, especially for dense topologies. Combining it with velocity re-initialization and local search, a minimum success rate of 0.9 is achieved in almost all the cases. It is also noticed that local search and velocity re-initialization save many iterations of computation.

Next, the effect of number of particles is investigated. The number of particles is varied from 20 to 120 and for each case, the success rate and the average best iteration number for the network topology # 2 in (from Table 1) to get the shortest path are recorded. Here, two cases are compared to highlight the effect of proposed encoding/decoding scheme. They are: (1) priority encoding with velocity re-initialization and local search and (2) proposed cost-biased encoding/decoding with velocity re-initialization and local search. The superiority of proposed encoding/decoding with velocity re-initialization and local search is clearly seen from the test results shown in Figs. 6 and 7.

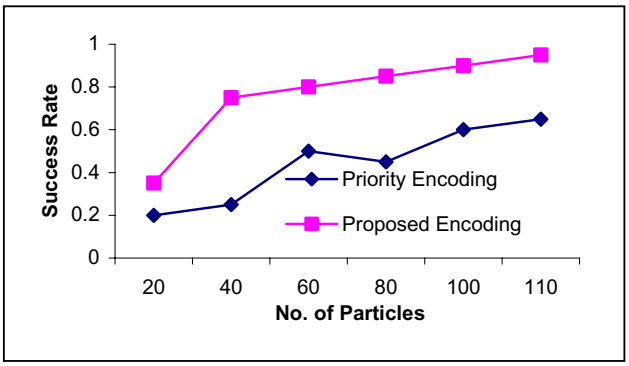

Fig. 6: Success rate vs. number of particles for a network of 100 nodes and 256 edges

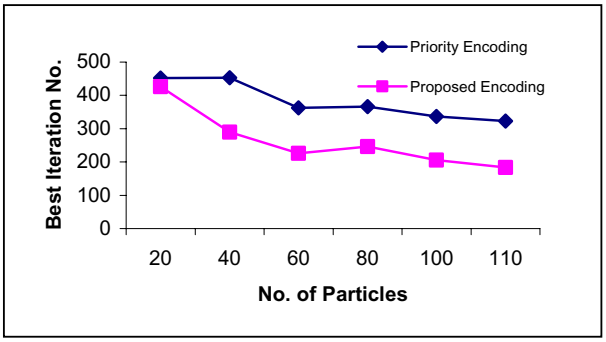

Fig. 7: Average iteration number to get the shortest path vs. number of particles for a network of 100 nodes and 256 edges

TABLE 1: SUCCESS RATES AND BEST ITERATIONS FOR DIFFERENT CASES OF PSO AND TRADITIONAL GA IMPLEMENTATIONS

Network Topology Number	Number of nodes	Number of edges	Priority Encoding in GA		Priority Encoding in PSO		Proposed Cost-biased encoding/decoding in PSO		Proposed Cost-biased encoding/decoding in PSO with local search and velocity re-initialization	
			Success rate	Average iteration No.	Success rate	Average iteration No.	Success rate	Average iteration No.	Success rate	Average iteration No.
1	100	290	0.3	260	0.4	289	0.7	241	0.9	180
2	100	256	0.45	272	0.65	258	0.8	243	0.95	206
3	90	253	0.3	263	0.35	265	0.7	223	1	149
4	90	233	0.45	233	0.6	218	0.85	175	1	132
5	80	230	0.6	147	0.6	228	0.9	209	1	131
6	80	186	0.65	158	0.5	176	0.9	162	1	125
7	70	320	0.3	157	0.4	213	0.65	212	0.9	149
8	70	202	0.6	152	0.7	274	0.9	151	0.9	127
9	60	233	0.55	181	0.45	235	0.8	225	0.95	102
10	50	158	0.75	156	0.8	137	1	94	1	60

V. CONCLUSIONS

This paper presents a PSO-based search algorithm for solving the single source shortest path problem. The algorithm uses a new cost-biased encoding/decoding for representation of particles in PSO so as to include the network parameters in the representation itself. In addition, for further enhancement of the search capability of the algorithm, selective local search and periodic velocity re-initialization are incorporated into the main PSO. The simulation experiments on random network topologies show that these two modifications (along with the new particle encoding/decoding) improve the performance of the algorithm significantly by achieving a success rate of more than 0.9 in all the case studies reported. The future work will focus on using PSO to solve other shortest path problem like the Steiner tree problem.

References:

[1] N. Deo, and C. Pang, ''Shortest-Path Algorithms: Taxonomy and Annotation,'' Networks, vol. 14, pp. 275-323, 1984.

[2] E. L. Lawler, Combinatorial Optimization: Networks and Matroids, New York; Holt, Rinehart and Winston, pp. 59 – 108, 1976.

[3] A. Chang Wook and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," IEEE Transactions on Evolutionary Computation, Vol. 6, No. 6, pp. 566-579, Dec. 2002.

[4] G. Mitsue, C. Runwei and D. Wang, "Genetic Algorithms for solving shortest path problems," Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 401 – 406, April 1997.

[5] G. Raidl, "A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem," Proc. SAC (1), pp.440–445, 2000.

[6] Z. Fu, A. Kurnia, A. Lim and B. Rodigrues, "Shortest path problem with cache dependent path lengths," Proceedings of 2003 Congress on Evolutionary Computation, pp. 2756 – 2761, 2003.

[7] J. Kuri, N. Puech, M. Gagnaire and E. Dotaro, "Routing foreseeable lightpath demands using a tabu search meta- heuristic," proceedings of 2003 Congress on Evolutionary Computation, pp. 2903 – 2807, 2003.

[8] J. Kennedy and R. C. Eberhart., "Particle swarm optimization.," Proceedings of the IEEE Int. Conf. on Neural Networks, Perth, Australia, pp. 1942-1948, 1995.

[9] Eberhart, R. C. and Shi, Y. Comparison between genetic algorithms and particle swarm optimization. Evolutionary programming vii: proc. 7th ann. conf. on evolutionary conf., Springer-Verlag, Berlin, San Diego, CA., 1998.

[10] C. R. Mouser and S. A. Dunn, "Comparing genetic algorithms and particle swarm optimization for an inverse problem exercise," ANZIAM Journal, 46(E), pp.C89-C101, 2005.

[11] D. W. Boeringer and D. H. Werner, "Particle swarm optimization versus genetic algorithms for phased array synthesis," Vol. 52, No. 3, pp. 771 – 779, 2004.

[12] Hassan R., Cohanim B., de Weck O.L., Venter G., "A Comparison of Particle Swarm Optimization and the Genetic Algorithm", AIAA-2005-1897, 1 st AIAA Multidisciplinary Design

Optimization Specialist Conference , Austin , Texas , April 18-21, 2005.

[13] C. Maurice, "The swarm and queen: Towards a deterministic and adaptive particle swarm optimization," Proceedings of the IEEE Congress on Evolutionary Computation. pp. 1951 – 1957, 1999.

[14] R. C. Eberhart and Y. Shi, "Comparing inertia weight and constriction factors in particle swarm optimization," Proceedings of the IEEE Congress on Evolutionary Computation., pp. 84-88, 2000.

[15] K. P. Wang, L. Huang, C. G. Zhou and W. Pang, "Particle swarm optimization for traveling salesman problem," Proceedings of International Conference on Machine Learning and Cybernetics, pp. 1583-1585, 2003.

[16] A. Salman, I. Ahmad, and S. Al-Madani, "Particle swarm optimization for task assignment problem," Microprocessors and Microsystems, vol. 26, no. 8, pp. 363-371, 2002.

[17] V. Kelner, F. Capitanescu, O. Léonard and L. Wehenkel, "An Hybrid Optimization Technique Coupling Evolutionary and Local Search Algorithms," Proceedings. of the 3rd International Conference on Advanced Computational Methods in Engineering (ACOMEN'2005), Ghent, Belgium,May--June 2005.

[18] T.A. Feo and M.G.C. Resende, "Greedy randomized adaptive search procedures," J. of Global Optimization, vol. 6, pp. 109-133, 1995.

[19] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: philosophy and performance difference," Proc. 7th Annual Conf. on Evolutionary Programming, pp. 601-610, 1998.

[20] P. Srinivas and Roberto Battiti, "The Gregarious Particle Swarm Optimizer (G-PSO)**,"** Proceedings of ACM Genetic and Evolutionary Computation Conference (GECCO) 2006, Seattle, Washington, USA, 2006 .

[21] X, Xie, W, Zang and Z Yang., "A dissipative swarm optimization," Proceedings of the IEEE Congress on Evolutionary Computing (CEC 2002*)*, Honolulu, Hawaii USA,May 2002.

[22] M. Iqbal, A. A. Freitas and C. G. Johnson, "Varying the topology and probability of re-initialization in particle swarm optimization.," In *Artificielle 2005,Edited by* E.-G. Talbi, *Evolution*. University of Lille, October 2005.

[23] B. Waxman, "Routing of multipoint connections," IEEE J. of Selected Areas in Communications*,* Vol. 6, No. 9, pp. 1622–1671, 1988.