Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2007, Article ID 27383, 25 pages
doi:10.1155/2007/27383

Research Article

Efficient Computation of Shortest Paths in Networks Using Particle Swarm Optimization and Noising Metaheuristics

Ammar W. Mohemmed and Nirod Chandra Sahoo

Received 13 March 2007; Accepted 4 April 2007

This paper presents a novel hybrid algorithm based on particle swarm optimization (PSO) and noising metaheuristics for solving the single-source shortest-path problem (SPP) commonly encountered in graph theory. This hybrid search process combines PSO for iteratively finding a population of better solutions and noising method for diversifying the search scheme to solve this problem. A new encoding/decoding scheme based on heuristics has been devised for representing the SPP parameters as a particle in PSO. Noising-method-based metaheuristics (noisy local search) have been incorporated in order to enhance the overall search efficiency. In particular, an iteration of the proposed hybrid algorithm consists of a standard PSO iteration and few trials of noising scheme applied to each better/improved particle for local search, where the neighborhood of each such particle is noisily explored with an elementary transformation of the particle so as to escape possible local minima and to diversify the search. Simulation results on several networks with random topologies are used to illustrate the efficiency of the proposed hybrid algorithm for shortest-path computation. The proposed algorithm can be used as a platform for solving other NP-hard SPPs.

1. Introduction

Shortest-path (SP) 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 [1–3], 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. Furthermore, the shortest-path problem also has numerous variations such as the minimum weight problem, the quickest path problem. Deo and Pang [4] have surveyed a large number of algorithms for and applications of the shortest-path problems.

The SPP has been investigated by many researchers. With the developments in communication, computer science, and transportation systems, more variants of the SPP have appeared. Some of these include traveling salesman problem, K-shortest paths, constrained shortest-path problem, multiobjective SPP, and network flow problems, and so forth. Most of these problems are NP-hard. Therefore, polynomial-time algorithms for these problems like Dijkstra and Bellman-Ford [5] are impossible. For example, in communication networks like IP, ATM, and optical network, there is a need to find a path with a minimum cost while maintaining a bound on delay to support quality-of-service applications. This problem is known to be NP-hard [6]. Multiple edge weights and weight limits may be defined, and the general problem is called as the constrained shortest-path problem. In another instance, it is required to find a shortest path such that cost or delay is to be minimized, and quality or bandwidth is to be maximized. These types of shortest paths are referred to as multicriteria or multiobjective shortest paths which are also NP-hard problems [6]. Therefore, untraditional methods like evolutionary techniques have been suggested to solve these problems which have the advantage of not only finding the optimal path, but also of finding suboptimal paths.

Artificial neural networks (ANNs) have been examined to solve the SP problem using their parallel and distributed architectures to provide a fast solution [7–9]. However, this approach has several limitations. These include the complexity of the hardware with increasing number of network nodes; at the same time, the reliability of the solution decreases. Secondly, they are less adaptable to topological changes in the network graph [9], including the cost of the arcs. Thirdly, the ANNs do not consider suboptimal paths. Thus, the evolutionary and heuristics algorithms are the most attractive alternative ways to go for. The powerful evolutionary programming techniques have considerable potential to be investigated in the pursuit for more efficient algorithms because the SP problem is basically an optimal search problem. In this direction, genetic algorithm (GA) has shown promising results [10–13]. The most recent notable results have been reported in [12, 13]. Their algorithm shows better performance compared to those of ANN approach and overcome the limitations mentioned above.

Among the notable evolutionary algorithms for path finding optimization problems in network graphs, successful use of GA and Tabu search (TS) has been reported [14–16]. The success of these evolutionary programming approaches promptly inspires investigative studies on the use of other similar (and possibly more powerful) evolutionary algorithms for this SP problem. Particle swarm optimization is one such evolutionary optimization technique [17], which can solve most of the problems solved by GA with less computational cost [18]. It is to be noted that GA and TS demand expensive computational cost. Some more comparative studies of the performances of GA and PSO have also been reported [19–22]. All these studies have firmly established similar effectiveness of PSO compared to GA. Even for some applications, it has been reported that the PSO performs better than other evolutionary optimization algorithms in terms of success rate and solution quality. The most attractive feature of PSO is that it requires less computational bookkeeping and, generally, a few lines of implementation codes. The basic philosophy and science behind PSO is based on the social behavior of a bird flock and a fish school, and so forth. Because of the specific algorithmic structure of PSO (updating of position and velocity of particles in a continuous manner), PSO has been mainly applied to many continuous optimization problems with few attempts for combinatorial optimization problems. Some of the combinatorial optimization problems that have been successfully solved using PSO are task assignment problem [23], traveling salesman problem [24, 25], sequencing problem [26], and permutation optimization problem [27], and so forth.

The purpose of this paper is to investigate on the applicability and efficiency of PSO for this SP problem. In this regard, this paper reports the use of particle swarm optimization to solve the shortest-path problem, where a new heuristics-based indirect encoding/decoding technique is used to represent the particle (position). This decoding/encoding makes use of cost of the edges and heuristically assigned node preference values in the network, that is, problem-specific. Further, in order to speed up the search process, additional noising metaheuristics [28, 29] have been fused into the iterative PSO process for effective local search around any better particle found in every PSO iteration. The basic purpose behind the use of this noising metaheuristics is as follows. Any effective search scheme for an optimization problem typically consists of two mechanisms/steps: (1) find a local optimal (better) solution, and (2) apply a diversification mechanism to find another local optimal solution starting from the one found in the previous step. The mechanism of PSO is very good in iteratively obtaining a population of local optimal solutions (in the end, one such solution may be the desired global solution) [17–27]. Recently, the noising method [28, 29] is shown to exhibit highly effective diversified local search. These features prompt the use of a hybrid search algorithm so as to exploit the good features of both PSO and noising method for solving the SPP. The proposed hybrid algorithm has been tested by exhaustive simulation experiments on various random network topologies. The analysis of the results indicates the superiority of the PSO-based approach over those using GA [12, 13].

The paper is organized as follows. In Section 2, standard PSO paradigm is briefly discussed. The proposed heuristics-based particle encoding/decoding mechanism is presented Section 3.2. In Section 3.4, the problem-specific local search algorithm using noising metaheuristics and the overall flow of the hybrid PSO algorithm for SP computation are shown. The results from computer simulation experiments are discussed in Section 4. Section 5 concludes the paper.

2. Basic particle swarm optimization algorithm

Particle swarm optimization is a population-based stochastic optimization tool inspired by social behavior of bird flock (and fish school, etc.), as developed by Kennedy and Eberhart [17]. This new evolutionary paradigm has grown in the past decade [30].

2.1. Basic steps of PSO algorithm. 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/generation in a specific manner as follows. In every iteration, the fitness of each particle is determined by some fitness measure and the velocity of each particle is updated by keeping track of the two "*best*" positions, that is, the first one is the best position (solution) a particle has traversed so far, called *pBest*, and the other "*best*" value is the best position that any neighbor of a particle has traversed so far, called neighborhood best (*nBest*). When a particle takes the whole population as its neighborhood, the neighborhood best becomes the global best and is accordingly called *gBest*. A particle's velocity and position are updated (till the convergence criterion, i.e., usually specified as maximum number of iterations, is met) as follows:

$$\mathrm{PV}_{id} = \mathrm{PV}_{id} + \varphi_1 r_1 (B_{id} - X_{id}) + \varphi_2 r_2 (B_{id}^n - X_{id}), \quad i = 1,2,\ldots,N_s,\ d = 1,2,\ldots,D, \tag{2.1}$$

$$X_{id} = X_{id} + \mathrm{PV}_{id}, \tag{2.2}$$

where φ_1 and φ_2 are positive constants, called as *acceleration coefficients*, N_s is the total number of particles in the swarm, D is the dimension of problem search space, that is, number of parameters of the function being optimized, r_1 and r_2 are independent random numbers in the range [0, 1], and "n" stands for the best neighbor of a particle. The other vectors are defined as $\mathbf{X}_i = [X_{i1}, X_{i2}, \ldots, X_{iD}] \equiv$ position of the ith particle; $\mathrm{PV}_i = [\mathrm{PV}_{i1}, \mathrm{PV}_{i2}, \ldots, \mathrm{PV}_{iD}]$ = velocity of the ith particle; $\mathbf{B}_i = [B_{i1}, B_{i2}, \ldots, B_{iD}]$ = best position of the ith particle ($pBest_i$), and $\mathbf{B}_i^n = [B_{i1}^n, B_{i2}^n, \ldots, B_{iD}^n]$ = best position found by the neighborhood of the particle i ($nBest_i$). When the convergence criterion is satisfied, the best particle found so far (with its position stored in $\mathbf{X}_{best}$ and best fitness f_{best}) is taken as the solution (near optimal) to the problem.

Equation (2.1) calculates a new velocity for each particle based on its previous velocity and present position, the particle's position at which the best possible fitness has been achieved so far, and the neighbors' best position achieved. Equation (2.2) updates each particle's position in the solution hyperspace. φ_1 and φ_2 are essentially 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 [17]. 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 (2.1). Thus, if the right-hand side of (2.1) exceeds a specified maximum value $\pm \mathrm{PV}_d^{\max}$, then the velocity on that dimension is clamped to $\pm \mathrm{PV}_d^{\max}$. Many improvements have been incorporated into this basic algorithm [31].

The commonly used PSOs are either global version of PSO or local version of PSO. In global version, all other particles influence the velocity of a particle, while in the local version of PSO, a selected number of neighbor particles affect the particle's velocity. In [32], PSO is tested with regular-shaped neighborhoods, such as global version, local version, pyramid structure, ring structure, and von Neumann topology. The neighborhood topology of the particle swarm has a significant effect on its ability to find optima. In ring topology, parts of the population that are distant from one another are also independent of one another. Influence spreads from neighbor to neighbor in this topology, until an optimum is found by any part of the population and then, this optimum will eventually pull all the particles into it. In the global version, every particle is connected to all other particles and influences all other particles immediately. The global populations tend to converge more rapidly than the ring populations, when they converge; but they are more susceptible to convergence towards local optima [33].

2.2. Modifications to basic PSO algorithm. For improved performance, some of the widely used modifications to the basic PSO algorithm are (a) constriction factor method, and (b) velocity reinitialization.

(a) *Constriction factor method (CFM).* In [34], Clerc proposed the use of a constriction factor χ. The algorithm was named the constriction factor method (CFM), where (2.1) is modified as

$$\mathrm{PV}_{id} = \chi\left[\mathrm{PV}_{id} + \varphi_1 r_1 (B_{id} - X_{id}) + \varphi_2 r_2 (B_{id}^n - X_{id})\right], \tag{2.3}$$

where

$$\chi = 2\left(\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|\right)^{-1} \quad \text{if } \varphi = \varphi_1 + \varphi_2 > 4. \tag{2.4}$$

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 [35] have reported that the best performance can be achieved with constriction factor and velocity clamping. Algorithm 2.1 shows pseudocodes of PSO (with CFM) for a function minimization problem. To pose a problem in PSO framework, the important step is to devise a coding scheme for particle representation, which is discussed in the following section.

(b) *Velocity reinitialization.* 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 [36]. 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 [37] avoids premature convergence of the swarm; the particles are reinitialized with a random velocity when stuck at a local minimum. Dissipative PSO [38] reinitializes the particle positions at each iteration with a small probability. In [39], this additional perturbation is carried out with different probabilities based on time-dependent strategy.

3. Shortest-path computation by PSO and noising metaheuristics

The shortest-path problem (SPP) 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 $(v_i, v_j, v_l, \ldots, v_k)$ in which no node is repeated. For example, in

$f_{best} \leftarrow \infty$, $\mathbf{X}_{best} \leftarrow NIL$
initialize $\mathbf{X}_i$ randomly
initialize PV_i randomly
evaluate fitness $f(\mathbf{X}_i)$
    $\mathbf{B}_i \leftarrow \mathbf{X}_i$
    $\mathbf{B}_i^n \leftarrow \mathbf{X}_j$ // j is the index of the best neighbor particle
iteration_count $\leftarrow 0$;
// *max_iteration = maximum number of iterations*

while (*iteration_count* < *max_iteration*)
    for each particle i,
        find $\mathbf{B}_i^n$ such that $f(\mathbf{B}_i^n) < f(\mathbf{X}_j)$, $\forall j \in$ {neighbors of i}
        if $f(\mathbf{X}_i) < f(\mathbf{B}_i)$ *then*
            $\mathbf{B}_i \leftarrow \mathbf{X}_i$
        if $f(\mathbf{B}_i) < f_{best}$ *then*
            $f_{best} \leftarrow f(\mathbf{B}_i)$
            $\mathbf{X}_{best} \leftarrow \mathbf{B}_i$
        update PV_i according to (2.3)
        update $\mathbf{X}_i$ according to (2.2)
        evaluate $f(\mathbf{X}_i)$
iteration_count $\leftarrow$ *iteration_count* + *1*;
end while
return $\mathbf{X}_{best}$

Algorithm 2.1. Simple particle swarm optimization algorithm (with CFM).

Figure 3.1, a path from node 1 to node 7 is represented as (1, 4, 2, 5, 7). The SPP is to find a path between two given nodes having minimum total cost. The 0-1 linear program model of the SPP is formulated as (s and t stand for source and terminal node, resp.):

$$\underset{h_{ij}}{\text{Minimize}} \sum_{(i,j)\in E} c_{ij}, \quad \text{such that} \sum_{j:(i,j)\in E} h_{ij} - \sum_{j:(j,i)\in E} h_{ji} = \begin{cases} 1, & i = s \\ -1, & i = t \\ 0, & i \neq s, t, \end{cases} \tag{3.1}$$

where h_{ij} is 1 if the edge connecting nodes i and j is in the path or 0 otherwise.

The proposed hybrid algorithm uses the PSO pseudocodes as listed in Algorithm 2.1 for network shortest-path computation with the inclusion of noising metaheuristics for diversified local search. In PSO, the quality of a particle (solution) is measured by a fitness function. For the SPP, the fitness function is obvious as the goal is to find the minimal cost path. Thus, the fitness of ith particle is defined as

$$f_i = \sum_{j=1}^{N_i-1} c_{yz}, \quad y = \mathbf{PP}^i(j),\ z = \mathbf{PP}^i(j+1), \tag{3.2}$$

where $\mathbf{PP}^i$ is the set of sequential node IDs for the ith particle, $N_i = |\mathbf{PP}_i|$ = number of nodes that constitute the path represented by the ith particle, and c_{yz} is the cost of the link

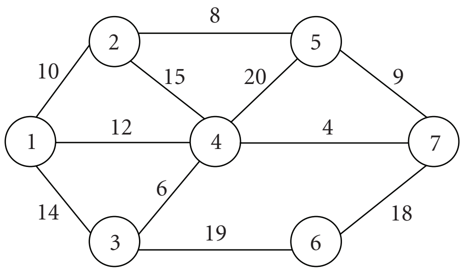

FIGURE 3.1. A network with 7 nodes and 11 edges.

connecting node y and node z. Thus, the fitness function takes minimum value when the shortest-path is obtained. If the path represented by a particle happens to be an invalid path, its fitness is assigned a penalty value so that the particle's attributes will not be considered by others for future search.

The main issue in applying PSO (GA) to the SPP is the encoding of a network path into a particle in PSO (chromosome in GA). This encoding in turn affects the effectiveness of a solution/search process. A brief discussion on some of the existing path encoding techniques for solving the SP problem using GA is presented followed by a detailed description of the proposed encoding algorithm.

3.1. Existing path encoding techniques. Two typical encoding techniques have been used for path representations in solving the SP problems using GA. They are direct and indirect representations. In the *direct representation scheme*, the chromosome in the GA is coded as a sequence of node identification numbers (node IDs) appearing in a path from a source node to a destination node [10–12]. A variable-length chromosome of length equal to the number of nodes for encoding the problem has been used to list up node IDs from a source node to a destination based on a topological database of a network. In [11], another similar (but slightly different) fixed-length chromosome representation has been used, that is, each gene in a chromosome represents a node ID that is selected randomly from the set of nodes connected with the node corresponding to its locus number. The disadvantage with these direct approaches is that a random sequence of node IDs may not correspond to a valid path (that terminates on destination node without any loop), increasing the number of invalid paths returned.

An *indirect scheme* for chromosome representation scheme has been proposed by Gen et al. [13], where instead of node IDs directly appearing on the path representation, some guiding information about the nodes that constitute the path is used to represent the path. The guiding information used in [13] are the *priorities* of various nodes in the network. During GA initialization, these priorities are assigned randomly. The path is generated by sequential node appending procedure beginning with the source node and terminating at the destination node, the procedure is referred as to path growth strategy. At each step of path construction from a chromosome, there are usually several nodes available for consideration and the one with the highest priority is added into path and the process is repeated until the destination node is reached. For effective decoding, a dynamic node adjacency matrix is maintained in the computer implementation [13] and is updated after every node selection so that a selected node is not a candidate for future selection. One main advantage of this encoding is that the size of the chromosome is fixed rather than being variable (as in direct encoding) making it easier to apply various operators like mutation and crossover. One disadvantage is that the chromosome is "indirectly" encoded; it does not have important information about the network's characteristics like its edges' costs. Actually this coding is quite similar to random number encoding used for graph tree representation in genetic algorithms [40].

Another variant of indirect coding of the chromosome is called *weighted encoding* [14]. 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 nonevolutionary *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.

3.2. Proposed cost-priority-based particle encoding/decoding. Inspired by the above encoding schemes, a representation scheme, called *cost-priority-based* encoding/decoding, is devised to suit the swarm particles for the SPP. Note that direct encoding is not appropriate for the particles as the particle updating uses arithmetic operations. In the proposed scheme, the particle encoding is based on node priorities and the decoding is based on the path growth procedure taking into account the node priorities as well as cost of the edges. The particle contains a vector of node priority values (particle length = number of nodes). To construct a path from a particle, from the initial node (node 1) to the final node (node n), the 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 priority (p_j) and the corresponding edge cost is minimum, that is,

$$j = \min\{c_{ij}p_j \mid (i,j) \in E\}, \quad p_j \in [-1.0, 1.0]. \tag{3.3}$$

The steps of this algorithm are summarized in Algorithm 3.1. The node priorities can take negative or positive real numbers in the range $[-1.0, 1.0]$. The problem parameters (edge costs) are part of the decoding procedure. Unlike the priority encoding where a node is appended to the partial path based only on its priority, in the proposed procedure, a node is appended to the path based on the minimum of the product of the node (next node) priority and the edge 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 priority values that result in shortest-path in a given network.

An example of the execution steps of the cost-priority decoding for path construction for the network of Figure 3.1 is shown in Figure 3.2. It also compares the path construction from the same particle with simple priority decoding [13] highlighting the advantage of the new approach.

```
// i is the source node
// j is an adjacent node to i
// n is the destination node, 1 = source node
// A(i) is the set of adjacent nodes to i
// PATH (k) is the partial path at decoding step k
// p_j is the corresponding priority of node j in the particle P (position vector X)
// N_∞ is a specified large number
Particle_Decoding (P)
i ← 1,
p_1 ← N_∞
k ← 0,
PATH (k) ← {1}
while ({j ∈ A(i), p_j ≠ N_∞} ≠ ∅)
      k ← k + 1
      j ← argmin{c_ij p_j | j ∈ A(i), p_j ≠ N_∞}
      i ← j, PATH (k) ← PATH(k) ∪ {i}
      p_i ← N_∞
      if  i = n  then return  the path PATH (k)
end while

return  Invalid_Path
```

Algorithm 3.1. Pseudocodes for cost-priority-based decoding procedure.

3.3. Noising metaheuristics-based local search for performance enhancement. Evolutionary algorithms (EAs) are robust and powerful global optimization techniques for solving large-scale problems with many local optima. However, they require high CPU times and are generally 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 [41, 42]. The hybridization of EA with local search has been shown to be faster and more promising on most problems. The local search essentially diversifies the search scheme. Recently, one such efficient metaheuristics, called *noising method*, was proposed by Charon and Hurdy [28, 29]. This noising metaheuristic was initially proposed for clique partitioning problem in a graph, and subsequently, it is shown to be very much successful for many combinatorial optimization problems.

In this work, a local search based on noising metaheuristics is embedded inside the main PSO algorithm for search diversification. The basic idea of noising method is as follows. for computation of the optimum of a combinatorial optimization problem, instead of taking the genuine data into account directly, they are perturbed by some progressively decreasing "*noise*" while applying local search. The reason behind the addition of noise is to be able to escape any possible local optimum in the optimizing function landscape. A noise is a value taken by a certain random variable following a given probability distribution (e.g., uniform or Gaussian law). There are different ways to add noise [29]. One

(a)

1	2	3	4	5	6	7
0.1	−0.4	0.7	0.6	0.3	0.5	0.2

$k = 0, PATH(k) = \{1\}$

$\Downarrow$

1	2	3	4	5	6	7
N_∞	−0.4	0.7	0.6	0.3	0.5	0.2

$(c_{12}p_2 = -4, c_{13}p_3 = 9.8), c_{14}p_4 = 7.2)$

$k = 1, PATH(k) = \{1,2\}$

$\Downarrow$

1	2	3	4	5	6	7
N_∞	N_∞	0.7	0.6	0.3	0.5	0.2

$(c_{24}p_4 = 9, c_{25}p_5 = 2.4)$

$k = 2, PATH(k) = \{1,2,5\}$

$\Downarrow$

1	2	3	4	5	6	7
N_∞	N_∞	0.7	0.6	N_∞	0.5	0.2

$(c_{57}p_7 = 1.8)$

$k = 3, PATH(k) = \{1,2,5,7\}$

Total path cost = 27

(a)

1	2	3	4	5	6	7
0.1	−0.4	0.7	0.6	0.3	0.5	0.2

$k = 0, PATH(k) = \{1\}$

$\Downarrow$

1	2	3	4	5	6	7
$-N_\infty$	−0.4	0.7	0.6	0.3	0.5	0.2

$k = 1, PATH(k) = \{1,3\}$

$\Downarrow$

1	2	3	4	5	6	7
$-N_\infty$	−0.4	$-N_\infty$	0.6	0.3	0.5	0.2

$k = 2, PATH(k) = \{1,3,4\}$

$\Downarrow$

1	2	3	4	5	6	7
$-N_\infty$	−0.4	$-N_\infty$	$-N_\infty$	0.3	0.5	0.2

$k = 3, PATH(k) = \{1,3,4,5\}$

$\Downarrow$

1	2	3	4	5	6	7
$-N_\infty$	−0.4	$-N_\infty$	$-N_\infty$	$-N_\infty$	0.5	0.2

$k = 4, PATH(k) = \{1,3,4,5,7\}$

Total path cost = 29

(b)

FIGURE 3.2. Illustrative examples of path construction from a particle position/priority vector for the network of Algorithm 2.1: (a) proposed cost-priority-based decoding; (b) simple priority-based decoding [13].

way is to add noise to the original data and then applying a descent search method on the noised data. The noising method used here is based on noising the *variations in the optimizing function* (f), that is, perturbing the variations of f. When a neighbor solution $\mathbf{X}'$ of the solution $\mathbf{X}$ is computed by applying an elementary transformation [28, 29] to $\mathbf{X}$, the genuine variation $\Delta f(\mathbf{X},\mathbf{X}')\{= f(\mathbf{X}') - f(\mathbf{X})\}$ is not considered, but a noised variation $\Delta f_{\text{noised}}(\mathbf{X},\mathbf{X}')$ defined by (3.4) is rather used:

$$\Delta f_{\text{noised}}(\mathbf{X},\mathbf{X}') = \Delta f(\mathbf{X},\mathbf{X}') + \rho_k, \tag{3.4}$$

where ρ_k denotes the noise (changing) at each trial k and depends on the noise rate (NR). Similar to iterative descent method in a function minimization problem, if $\Delta f_{\text{noised}}(\mathbf{X}, \mathbf{X}') < 0$, $\mathbf{X}'$ becomes the new current solution, otherwise $\mathbf{X}$ is kept as the current solution and another neighbor of $\mathbf{X}$ is tried. The noise ρ_k is randomly chosen from an interval whose range decreases during the process (typically to zero, but it is often stopped much earlier). For example, if noise is drawn from the interval $[-NR, +NR]$ with a given probability distribution, then the noise rate NR of the noise decreases during the running of the method from $NR_{\max}$ to $NR_{\min}$, as given in (3.5), used in this study. The noise is added in an interval containing positive as well as negative values such that a bad neighboring

```
// X is the initial solution
// X′ is a neighbor of X computed by an elementary transformation on X
// better_sol is the best solution found so far
// NR is the noise rate
// ρ_k is a random real number
// add_noise is a logical variable used to choose between noised or unnoised
descent (local search)
Noising_Method (X)
k ← 0
add_noise ← false
better_sol ← X
NR ← NR_max
While (k < max_trials)

    if (k = 0 (modulo fixed_rate_trials) and add_noise = false) then
    // noised descent phase
            NR = NR_max(1 − k/max_trials)
            ρ_k ← uniformly drawn from [−NR, NR]
            add_noise = true

    else if (k = 0 (modulo fixed_rate_trials) and add_noise = true) then
    // unnoised descent phase
            ρ_k ← 0
            add_noise = false
     Let X′ be the next neighbor of X.
     if f(X′) − f(X) + ρ_k < 0, then X ← X′.
     if f(X) < f(better_sol), then better_sol ← X.
k ← k + 1

end while
return better_sol
```

Algorithm 3.2. Pseudocodes for local search using noising metaheuristics.

solution may be accepted, but also a good neighboring solution may be rejected,

$$NR = NR_{\max} \times \left(1 - \frac{k}{\text{max\_trials}}\right), \tag{3.5}$$

where k = trial number in the noising-method-based local search, max_trials = total number of trials for a typical current solution, and fixed_rate_trials = number of trials with fixed noise rate. We follow [29] where noising method is applied alternatively with unnoised (descent) search, that is, for few trials, noised descent is performed followed by unnoised descent search for few trials and so on. The pseudocodes for the noising-method-based local search for an initial solution **X** are given in Algorithm 3.2.

3.4. Complete PSO and noising-method-based algorithm for SPP. The complete algorithm for the shortest-path computation uses main PSO algorithm (Figure 3.1) with an embedded selective local search done on each particle using noising metaheuristics as discussed above. The local search is performed selectively making use of the concept of proximate optimality principle (POP) [43]. It has been experimentally shown that the POP holds good for many combinatorial optimization problems, that is, good solutions in optimization problems have similar structures. The good solutions are interpreted as locally optimal solutions as obtained from the main PSO. Based on POP, a good solution (a path in the SPP) is more likely to have better solutions in its locality, that is, another better solution (path) in the local neighborhood most likely shares some node/edges (similar structures). Conversely, it is not advised to have a local search around a known bad solution. This feature is incorporated in the proposed hybrid algorithm by applying nosing-method-based local search only when a solution's fitness improves by PSO (then one may expect to get locally better solutions). The pseudocodes for the complete hybrid algorithm for the SPP are given in Algorithm 3.3.

The algorithm passes the particle that has experienced an improvement in PSO to the noising method. The noising method will take this particle as an initial solution for further search around it. If the noising local search is able to find a solution better than the original particle, then the particle will be updated and returned. Also, this new solution is compared with best solution found so far by that particle; if it is better, then it will also be updated for reflecting the new found solution back on the swarm.

4. Computer simulation results and discussion

The proposed PSO-based hybrid algorithm for SPP is tested on networks with random and varying (The random network topologies are generated using Waxman model [44] in which nodes are generated randomly on a two-dimensional plane of size 100×100, and there is a link between two nodes u and v with probability $p(u,v) = \boldsymbol{\alpha} \cdot e^{-d(u,v)/(\beta L)}$, here $0 < \alpha, \beta \leq 1$, $d(u,v)$ is the Euclidean distance between u and v, and L is the maximum distance between any two nodes) topologies through computer simulations using Microsoft Visual C++ on an Pentium 4 processor with 256 MB RAM. The edge costs of the networks are randomly chosen in the interval [10, 1000]. The results are also compared with two recently reported GA-based approaches, that is, one uses direct encoding scheme [12] and the other uses the priority-vector-based indirect encoding scheme (but without the modifications proposed in this work) [13]. In all the simulation tests, the optimal solution obtained using Dijkstra's algorithm [4, 5] is used as reference for comparison purposes. The selections of parameter settings of PSO and noising metaheuristics are discussed now.

(a) *Population size*. In general, any evolutionary search algorithm shows improved performance with relatively larger population. However, very large population size means greater cost in terms of fitness function evaluations. In [45, 46], it is stated that a population size of 30 is a reasonably good choice.

(b) *Particle initialization*. The particle position (node priorities) and velocity are initialized with random real numbers in the range $[-1.0, 1.0]$. The maximum velocity is set as ± 1.0.

```
f_best ← ∞
X_best ← NIL
PATH ← NIL
for each particle i,
        initialize X_i randomly from [−1.0, 1.0]
        initialize PV_i randomly from [−1.0, 1.0]
        evaluate f(X_i)
            PATH ← Particle_Decoding(X_i)
            f(X_i) ← cost (PATH)
        B_i ← X_i
        B_i^n ← X_j // j is the index of the best neighbor particle

iteration_count ← 0;
// max_iteration is the specified maximum number of iterations
while (iteration_count < max_iteration)

      for each particle i
           find B_i^n such that f(B_i^n) < f(X_i)
           if f(X_i) < f(B_i) then
                        B_i ← X_i
                        B_i ← Noising_Method(B_i)
           if f(B_i) < f_best then
              f_best ← f(B_i)
              X_best ← B_i
           update PV_i according to (2.3)
           update X_i according to (2.2)
           evaluate f(X_i)
              PATH ← Particle_Decoding(X_i)
                f(X_i) ← cost (PATH)
iteration_count ← iteration_count + 1

end while

PATH ← Particle_Decoding(X_best)
return PATH
```

ALGORITHM 3.3. Pseudo-codes for hybrid PSO (with CFM) and noising metaheuristics for the SPP.

(c) *Neighborhood topology*. Ring neighborhood topology [33] for PSO is used to avoid premature convergence. In this topology, each particle is connected to its two immediate neighbors.

(d) *Constriction factor* χ. In [47], it is shown that the CFM has linear convergence for $\varphi > 4$. Here, φ_1 and φ_2 are chosen to be 2 and 2.2, respectively; thus $\varphi = 4.2$. From (2.4), $\chi = 0.74$.

(e) *Noising method parameters*. Maximum and minimum noise rates $NR_{\max} = 80$, $NR_{\min} = 0$; maximum number of trials (*max_trials*) = 4000; maximum number of trials at a fixed noise rate (*fixed_rate_trials*) = 10. The generic elementary transformation used for local neighborhood search is the swapping of node priority values at two randomly selected positions of a particle priority (position) vector and two such swapping transformations are successively applied in each trial for generating a trial solution in the local search.

4.1. Performance assessment of proposed hybrid PSO. The main objective of these simulation experiments is to investigate the quality of solution and convergence speed for different network topologies. First, the quality of solution (route optimality) is investigated. The route optimality (or success rate) is defined as the (average number) percentage of times the algorithm finds the global optimum (i.e., the shortest-path) [12] over a large number of runs. The route failure ratio is the inverse of route optimality. It is asymptotically the probability that the computed route is not optimal, as it is the relative frequency of route failure [12]. The results are averaged over 1000 runs and in each run (for a network of certain number of nodes), a different random network is generated by changing the seed number. The seed number changes from 1 to 1000 generating networks with a minimum degree of 4 and maximum degree of 10. The number of fitness function evaluations to achieve the corresponding success rate is also recorded. In all the cases, the proposed cost-priority-based encoding/decoding of particle is used.

Case 1: standard PSO (with CFM only).
Case 2: standard PSO (with CFM and velocity reinitialization).
Case 3: noising method (where PSO is used for initial two iterations to obtain good starting points).
Case 4: standard PSO (with CFM) and local search with noising method (*proposed hybrid algorithm*).

For all cases, the number of particles is chosen to be 30. Maximum number of PSO iterations is chosen as 100 except for the cases where no noising-method-based local search is used, that is, for Cases 1 and 2, the maximum number of iterations is chosen as 2000 so that a fair comparison of performance is done because with the noising method, more local search trials are being performed. The number of noising trials is set to 4000. For Case 3, two initial PSO iterations are used to generate a better initial solution and then the noising method is allowed to run for 30 000 trials.

Table 4.1 shows a comparison of success rates (SRs) and required average number of fitness function evaluations (FE_{av}) for convergence to the reported results. It is seen that Case 2 which adopts velocity reinitialization shows better results over Case 1 which implements the standard PSO only. Case 3 has the worst performance and the algorithm fails to give a success rate more than 66%. The proposed hybrid PSO algorithm (Case 4) that incorporates noising-method-based local search offers the best results in terms of success rate as well as convergence speed. The proposed algorithm is clearly able to find the optimum path with high probability (more than 95%) for most of the network topologies tested.

Next, the effect of the number of particles is investigated. For this, two case studies (Cases 2 and 4) are selected. The number of particles is varied from 10 to 50, and for each case, the success rate and the average number of fitness function evaluations required to achieve that success rate for the network topology number 1 (from Table 4.1)

Table 4.1. Performance comparison among different case studies involving PSO for the SPP.

Network topology number	No. of nodes	No. of edges	Case 1		Case 2		Case 3		Case 4	
			SR	FE_{av}	SR	FE_{av}	SR	FE_{av}	SR	FE_{av}
1	100	281	0.656	35961	0.637	32800	0.453	19750	0.957	22858
2	100	255	0.634	32270	0.686	29469	0.463	19872	0.966	17648
3	90	249	0.654	30713	0.694	28393	0.545	17723	0.971	17210
4	90	227	0.695	27632	0.768	24125	0.546	17622	0.982	13993
5	80	231	0.691	28444	0.733	25372	0.553	17364	0.982	16155
6	80	187	0.804	20354	0.821	18816	0.632	14973	0.984	10035
7	70	321	0.498	39352	0.58	34789	0.363	22119	0.897	34222
8	70	211	0.724	25277	0.771	22438	0.57	16884	0.973	15516
9	60	232	0.681	27924	0.742	24352	0.496	18541	0.961	20269
10	50	159	0.85	15304	0.872	13475	0.662	13328	0.995	8107

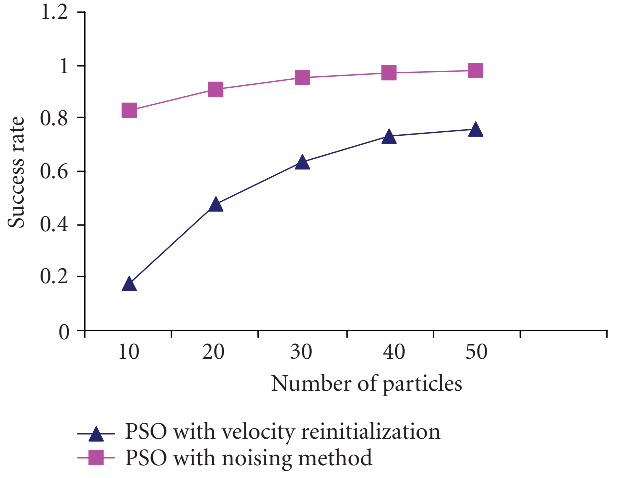

Figure 4.1. Success rate versus number of particles for network number 1.

are recorded. From the comparison of test results shown in Figures 4.1 and 4.2, it is seen that the proposed hybrid algorithm based on PSO and noising method produces better results for all the different population settings. For example, with number of particles equal to 30, the success rate with proposed algorithm is 95.7% (compared to 63.7% for the case without noising method), and for number of particles equal to 40, the success rate with proposed algorithm is 97% (compared to 73.3% for the case without noising method).

Further, the effects of the number of PSO iterations and noising-method-based trials on convergence characteristics of the algorithm are examined (with a population size

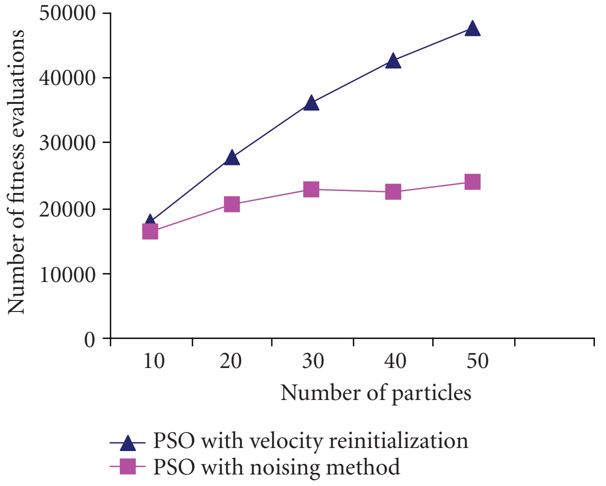

FIGURE 4.2. Average number of fitness function evaluations to get the results of Figure 4.1.

of 30). In the first experiment (Case A), the number of noising method trials is fixed to 1000 and the number of PSO iterations is changed from 20 to 100. The success rate and number of fitness function evaluations are recorded. In the second experiment (Case B), the number of PSO iterations is fixed to 10 and the number of noising method trials is changed from 1000 to 5000. These specific choices are set to get a comparison of success rates and number of fitness function evaluations for the two cases, that is, variable number of PSO iterations and variable number of noising method trials. The results are illustrated in Figures 4.3 and 4.4. As expected, to get high success rates, either the number of PSO iterations should be increased when the proposed algorithm is used with certain fixed number of noising-method-based trials or the other way round. However, it should be noted that a noising-method-based trial is simpler in terms of computation demand compared to PSO iteration. Hence, the algorithm is more responsive to the increase in the number of noising method trials than PSO iterations. Increasing the number of PSO iterations while keeping the number of noising method trials fixed has less performance efficiency than increasing number of noising trials for fixed number of PSO iterations. For example, with 10 PSO iterations and 5000 noising-method-based trials, the success rate is 88% and the average number of fitness function evaluations is 17570 compared to getting a success rate of 75% and 17962 fitness function evaluations for the case with 100 PSO iterations and 1000 noising-method-based trials.

An advantage of the proposed algorithm is that several alternative suboptimal paths are also generated in the search process. This is recorded in Figure 4.5 which shows the success rates for suboptimal paths with costs within 105% and 110% of the shortest-path cost. The success rates are almost 100% for all the topologies. Figure 4.6 shows the number of unique paths (generated in the search process) whose costs are within 115% of the optimum path cost. The point is that the proposed algorithm also successfully generates many potential paths between the source and the destination.

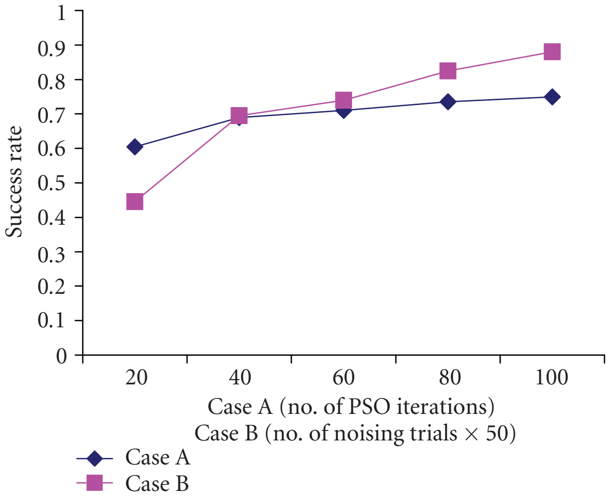

FIGURE 4.3. Success rate versus number of iterations for network number 1.

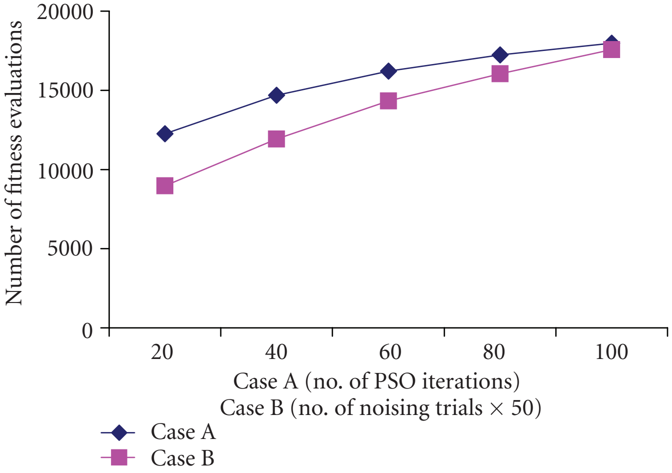

FIGURE 4.4. Number of evaluations versus number of iterations for network number 1.

All these results clearly establish the superiority of the nosing method based hybrid PSO algorithm for solving the shortest-path problem in terms of solution quality as well as convergence characteristics. Further, in order to assess the relative performance of the proposed algorithm for the SPP compared to other previously reported heuristic algorithms, in the following two subsections, the simulation results are also compared with those obtained from other previously reported results for this problem, that is, GA-based search using direct path encoding scheme [12] and indirect (priority-based) path encoding scheme [13].

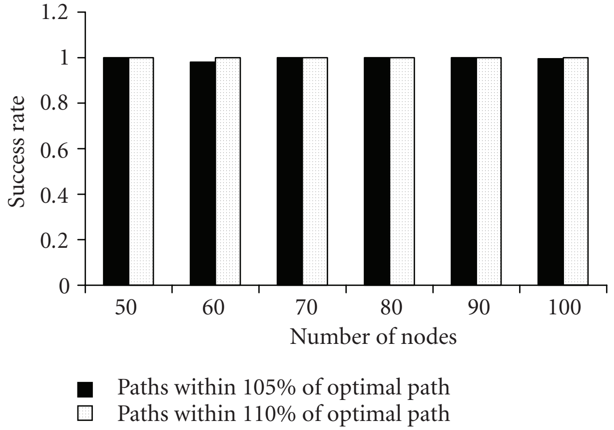

FIGURE 4.5. Success rate for alternative paths that are within 105% and 110% of the optimum path cost.

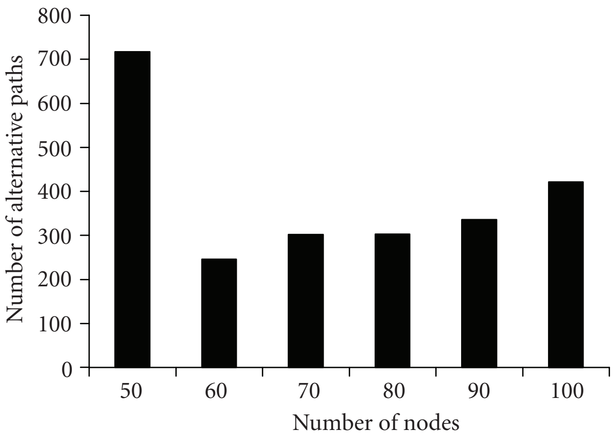

FIGURE 4.6. Number of alternative paths that are within 115% of the optimum path cost.

4.2. Performance comparison with GA-based search using direct path encoding. To compare the performance of the proposed PSO hybrid algorithm with the GA algorithm described in [12], different network topologies of (15–50) nodes with randomly assigned link are generated. A total of 1000 random network topologies was considered in each case (number of nodes). The number of PSO iterations is set to 100 and the number of noising method trials is 4000. A comparison of the quality of solution in terms of route failure ratio between the proposed PSO-based hybrid algorithm and GA-based search reported in [12] (where the number of chromosomes in each case is the same as the number of nodes in the network) is shown in Figures 4.7 and 4.8 comparing the time efficiency to get these results (same hardware setup as in [12]). It clearly illustrates that the quality of

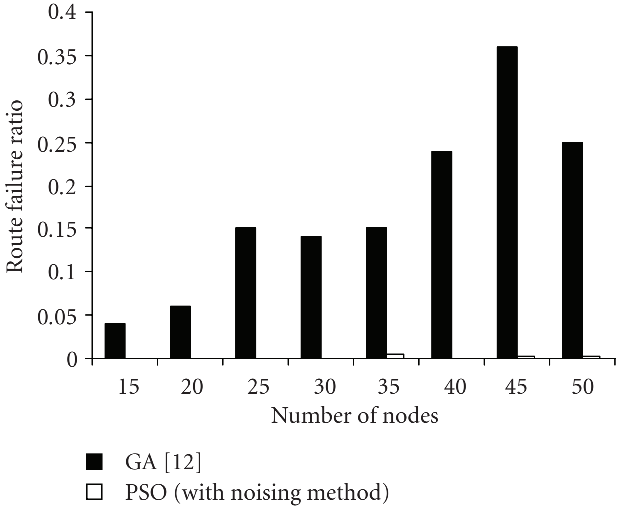

Figure 4.7. Comparison of route failure ratio between proposed hybrid PSO and GA [12].

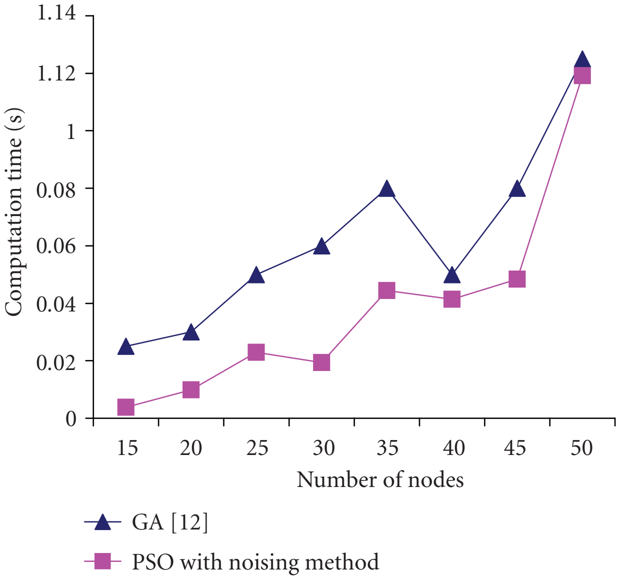

Figure 4.8. Comparison of convergence time between proposed hybrid PSO and GA [12].

solution and time efficiency obtained with PSO-based hybrid algorithm are higher than those of GA-based search. For example, in case of 45 node networks, the route failure ratio is 0.002 (99.8% route optimality); but the GA search has route failure ratio 0.36 (64% route optimality). The overall statistics of these results are collected in Table 4.2. The

Table 4.2. Comparison of statistics of the quality of solution.

Performance measure		Algorithms	
		GA search [12]	PSO-based search (proposed)
Route failure ratio	Average	0.1712	0.0015
	standard deviation	0.1067	0.0016

Table 4.3. Different testing conditions for test number 1.

Case study	Network used in [13]: (nodes, edges, optimal path cost)	Network used here: (nodes, edges, optimal path cost)	Population size	Number of generations used in [13]	Number of PSO iterations/noising method trials
I	(6,10,10)	(6, 10,11)	10	100	10/1000
II	(32,66,205)	(32, 66,228)	20	200	10/1000
III	(70,211,2708)	(70,224,2780)	40	400	10/1000

PSO-based search attains an average route failure ratio of 0.0015 (99.85% route optimality) compared to 0.1712 for the GA search [12]. The standard deviation of route failure ratio for the proposed PSO-based hybrid algorithm amounts 0.0016 compared to 0.1067 for GA search. Clearly, the proposed algorithm outperforms the GA-based algorithm for this problem.

4.3. Performance comparison with GA-based search using indirect path encoding. For performance comparison of PSO-based hybrid search algorithm, that is, PSO and noising-method-based local search, using proposed encoding/decoding technique with those reported in [13] using GA-based search and indirect encoding scheme, the same testing conditions are simulated. However, [13] only reports the number of nodes and edges in all the used networks and no information on the cost of the edges is provided. Thus, the closest possible network is generated in this study where the number of nodes of each network is exactly the same used in [13] and the number of edges is as close as possible to those of [13]. The results are summarized as follows.

(a) *Test number 1.* Three random networks of different sizes are generated. The testing conditions are given in Table 4.3. The statistical results for frequency of obtaining optimal path over 400 independent runs (with different seeds) are compared in Table 4.4. Clearly, the proposed PSO-based search performs better.

(b) *Test number 2.* In this test, the effects of population size on convergence characteristics are compared. The number of generations/iterations in every run is fixed. The testing conditions are number of PSO iterations = 10 (200 iterations in [13]), the chosen network is that mentioned for case study III (only) in Table 4.3 for respective algorithms, and the population sizes for both are varied from 10 to 100. The comparisons of frequency for obtaining optimal path obtained from 200 random runs (for each population

TABLE 4.4. Comparison of statistical results between GA-based search [13] and proposed algorithm for test number 1.

Case study	Frequency for obtaining the optimal path	
	GA-based search [13]	Proposed PSO-based hybrid algorithm
I	100%	100%
II	98%	100%
III	64%	99%

TABLE 4.5. Comparison of statistical results between GA-based search [13] and proposed algorithm for test number 2.

Population size	Frequency for obtaining the optimal path	
	GA-based search [13]	Proposed PSO-based hybrid algorithm
10	21%	55%
40	64%	93%
60	83%	98%
100	92%	100%

TABLE 4.6. Comparison of statistical results between GA-based search [13] and proposed algorithm for test number 3.

Number of generations/iterations	Frequency for obtaining optimal path	
	GA-based search [13] (population size = 10)	Proposed PSO-based hybrid algorithm
100	10%	55%
400	42%	95%
800	66%	97%
1200	76%	98%
2000	92%	99%
3000	94%	99%

size) are summarized in Table 4.5. As anticipated, the frequency for obtaining optimal solution increases with population size for both approaches. The superior performance of the proposed PSO-based hybrid search algorithm is again highlighted in the results.

(c) *Test number 3.* In this test, the convergence characteristics of search algorithms are compared when the population size is fixed and the number of generations/iterations is gradually increased from 100 to 3000. The chosen networks for both the search algorithms are again the respective networks given in case study III in Table 4.3. The results over 200 runs for each case are summarized in Table 4.6.

4.4. Overall remarks on performance of the proposed algorithm. In general, the results show that the hybridization of PSO with the noising-method-based local search improves the overall performance. Using pure PSO while increasing iterations or population size does not improve performance and it is also computationally expensive. In the proposed hybrid PSO algorithm, the number of PSO iterations and the number of noising method trials play a role in improving performance and reducing computation time. Generally, PSO iteration is more expensive than noising method trial. Also, PSO iterations involve velocity and position updating for the whole population, while noising method trial involves only an elementary transformation and updating a new (better) solution found for one particle. Therefore, a balance between the two schemes is necessary to get better quality of solution with reasonable computation-time efficiency. A population size of 30 with $80(-100)$ PSO iterations and 4000 noising-method-based local search trials seems to give good results in reasonable time for the complex network of 100 nodes and 281 edges. If a near-optimum solution is enough for smaller-size networks, then less number of iterations can be used.

5. Conclusions

In this paper, a hybrid PSO/noising method algorithm is presented and tested for solving the shortest-path problem in networks. A new cost-priority-based particle encoding/decoding scheme has also been devised so as to incorporate the network-specific heuristic information in the path construction process. The simulation results on a wide variety of random networks show that the proposed algorithm produces good results in terms of higher success rates for getting the optimal path, which is also better than those reported in the literature for the shortest-path problem using GA-based search algorithms. In hybrid techniques, one technique can be used to overcome the disadvantage of the other. It is believed that there is still a room for improving the performance of the algorithm by adding more techniques like the Tabu search. The advantage of this proposed heuristic algorithm is that it can be easily extended to solve the other variants of the shortest-path problems like the constrained shortest path, multicriteria shortest path, and so forth, which are known to be NP-hard and no polynomial-time solution is known for them. This is under investigation in our future work.

References

[1] F. B. Zahn and C. E. Noon, "Shortest path algorithms: an evaluation using real road networks," *Transportation Science*, vol. 32, no. 1, pp. 65–73, 1998.

[2] J. Moy, "Open shortest path first Version 2. RFQ 1583," Internet Engineering Task Force, 1994 http://www.ietf.org/.

[3] G. Desaulniers and F. Soumis, "An efficient algorithm to find a shortest path for a car-like robot," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 6, pp. 819–828, 1995.

[4] N. Deo and C. Y. Pang, "Shortest-path algorithms: taxonomy and annotation," *Networks*, vol. 14, no. 2, pp. 275–323, 1984.

[5] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, NY, USA, 1976.

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, Calif, USA, 1979.

[7] M. K. M. Ali and F. Kamoun, "Neural networks for shortest path computation and routing in computer networks," *IEEE Transactions on Neural Networks*, vol. 4, no. 6, pp. 941–954, 1993.

[8] J. Wang, "A recurrent neural network for solving the shortest path problem," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '94)*, vol. 6, pp. 319–322, London, UK, May-June 1994.

[9] F. Araujo, B. Ribeiro, and L. Rodrigues, "A neural network for shortest path computation," *IEEE Transactions on Neural Networks*, vol. 12, no. 5, pp. 1067–1073, 2001.

[10] M. Munemoto, Y. Takai, and Y. Sato, "A migration scheme for the genetic adaptive routing algorithm," in *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics*, vol. 3, pp. 2774–2779, San Diego, Calif, USA, October 1998.

[11] J. Inagaki, M. Haseyama, and H. Kitajima, "A genetic algorithm for determining multiple routes and its applications," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '99)*, vol. 6, pp. 137–140, Orlando, Fla, USA, May-June 1999.

[12] C. W. Ahn 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, 2002.

[13] M. Gen, R. Cheng, and D. Wang, "Genetic algorithms for solving shortest path problems," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 401–406, Indianapolis, Ind, USA, April 1997.

[14] G. Raidl and B. A. Julstrom, "A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem," in *Proceedings of the ACM Symposium on Applied Computing (SAC '00)*, vol. 1, pp. 440–445, Como, Italy, March 2000.

[15] Z. Fu, A. Kurnia, A. Lim, and B. Rodrigues, "Shortest path problem with cache dependent path lengths," in *Proceedings of the Congress on Evolutionary Computation (CEC '03)*, vol. 4, pp. 2756–2761, Canberra, Australia, December 2003.

[16] J. Kuri, N. Puech, M. Gagnaire, and E. Dotaro, "Routing foreseeable light path demands using a tabu search meta-heuristic," in *Proceedings of IEEE Global Telecommunication Conference (GLOBECOM '02)*, vol. 3, pp. 2803–2807, Taipei, Taiwan, November 2002.

[17] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Western Australia, November-December 1995.

[18] R. Hassan, B. Cohanim, O. L. DeWeck, and G. Venter, "A comparison of particle swarm optimization and the genetic algorithm," in *Proceedings of the 1st AIAA Multidisciplinary Design Optimization Specialist Conference*, Austin, Tex, USA, April 2005.

[19] E. Elbeltagi, T. Hegazy, and D. Grierson, "Comparison among five evolutionary-based optimization algorithms," *Advanced Engineering Informatics*, vol. 19, no. 1, pp. 43–53, 2005.

[20] C. R. Mouser and S. A. Dunn, "Comparing genetic algorithms and particle swarm optimization for an inverse problem exercise," *The Australian & New Zealand Industrial and Applied Mathematics Journal*, vol. 46, part C, pp. C89–C101, 2005.

[21] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Proceedings of the 7th International Conference on Evolutionary Programming*, pp. 611–616, Springer, San Diego, Calif, USA, March 1998.

[22] D. W. Boeringer and D. H. Werner, "Particle swarm optimization versus genetic algorithms for phased array synthesis," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 3, pp. 771–779, 2004.

[23] 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.

[24] K.-P. Wang, L. Huang, C.-G. Zhou, and W. Pang, "Particle swarm optimization for traveling salesman problem," in *Proceedings of the 2nd International Conference on Machine Learning and Cybernetics (ICMLC '03)*, vol. 3, pp. 1583–1585, Xi'an, China, November 2003.

[25] M. Clerc, "Discrete particle swarm optimization illustrated by the traveling salesman problem," 2000, http://www.mauriceclerc.net/.

[26] L. Cagnina, S. Esquivel, and R. Gallard, "Particle swarm optimization for sequencing problem: a case study," in *Proceedings of the IEEE Conference on Evolutionary Computation (CEC '04)*, vol. 1, pp. 536–541, Portland, Ore, USA, June 2004.

[27] X. Hu, R. C. Eberhart, and Y. Shi, "Swarm intelligence for permutation optimization: a case study of n-queens problem," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '03)*, pp. 243–246, Indianapolis, Ind, USA, April 2003.

[28] I. Charon and O. Hudry, "The noising method: a new method for combinatorial optimization," *Operations Research Letters*, vol. 14, no. 3, pp. 133–137, 1993.

[29] I. Charon and O. Hurdy, "The noising methods: a generalization of some metaheuristics," *European Journal of Operational Research*, vol. 135, no. 1, pp. 86–101, 2001.

[30] X. Hu, Y. Shi, and R. C. Eberhart, "Recent advances in particle swarm," in *Proceedings of the Congress on Evolutionary Computation (CEC '04)*, vol. 1, pp. 90–97, Portland, Ore, USA, June 2004.

[31] Y. Shi, "Particle swarm optimization," Feature article, IEEE Neural Networks Society, February 2004.

[32] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, vol. 2, pp. 1671–1676, Honolulu, Hawaii, USA, May 2002.

[33] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, vol. 3, pp. 1931–1938, Washington, DC, USA, July 1999.

[34] M. Clerc, "The swarm and queen: towards a deterministic and adaptive particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, vol. 3, pp. 1951–1957, Washington, DC, USA, July 1999.

[35] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '00)*, vol. 1, pp. 84–88, La Jolla, Calif, USA, July 2000.

[36] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: philosophy and performance difference," in *Proceedings of the 7th International Conference on Evolutionary Programming*, pp. 601–610, San Diego, Calif, USA, March 1998.

[37] P. Srinivas and R. Battiti, "The gregarious particle swarm optimizer (G-PSO)," in *Proceedings of the 8th Annual Conference Genetic and Evolutionary Computation (GECCO '06)*, pp. 67–74, Seattle, Wash, USA, July 2006.

[38] X.-F. Xie, W.-J. Zang, and Z.-L. Yang, "Dissipative particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, vol. 2, pp. 1456–1461, Honolulu, Hawaii, USA, May 2002.

[39] M. Iqbal, A. A. Freitas, and C. G. Johnson, "Varying the topology and probability of re-initialization in particle swarm optimization," in *Proceedings of the 7th International Conference on Artificial Evolution*, Lille, France, October 2005.

[40] F. Rothlauf, D. E. Goldberg, and A. Heinzl, "Network random keys: a tree network representation scheme for genetic and evolutionary algorithms," *Evolutionary Computation*, vol. 10, no. 1, pp. 75–97, 2002.

[41] V. Kelner, F. Capitanescu, O. Léonard, and L. Wehenkel, "A hybrid optimization technique coupling evolutionary and local search algorithms," in *Proceedings of the 3rd International Conference on Advanced Computational Methods in Engineering (ACOMEN '05)*, Ghent, Belgium, May-June 2005.

[42] T. A. Feo and M. G. C. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[43] K. Yasuda and T. Kanazawa, "Proximate optimality principle based Tabu search," in *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, vol. 2, pp. 1560–1565, October 2003.

[44] B. M. Waxman, "Routing of multipoint connections," *Journal of Selected Areas in Communications*, vol. 6, no. 9, pp. 1617–1622, 1988.

[45] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, vol. 3, pp. 1945–1950, Washington, DC, USA, July 1999.

[46] A. Carlisle and G. Dozier, "An off-the-shelf PSO," in *Proceedings of the Workshop on Particle Swarm Optimization*, pp. 1–6, Indianapolis, Ind, USA, April 2001.

[47] M. Clerc and J. Kennedy, "The particle swarm explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

Ammar W. Mohemmed: Faculty of Engineering and Technology, Multimedia University,
Jalan Ayer Keroh Lama, Melaka 75450, Malaysia
Email address: ammar.wmohemmed@mmu.edu.my

Nirod Chandra Sahoo: Faculty of Engineering and Technology, Multimedia University,
Jalan Ayer Keroh Lama, Melaka 75450, Malaysia
Email address: nirodchandra.sahoo@mmu.edu.my