Noname manuscript No.
(will be inserted by the editor)

A Fuzzy Particle Swarm Optimization Algorithm for Computer Communication Network Topology Design

Salman A. Khan · Andries P. Engelbrecht

Received: date / Accepted: date

Abstract Particle swarm optimization (PSO) is a powerful optimization technique that has been applied to solve a number of complex optimization problems. One such optimization problem is topology design of distributed local area networks (DLANs). The problem is defined as a multi-objective optimization problem requiring simultaneous optimization of monetary cost, average network delay, hop count between communicating nodes, and reliability under a set of constraints. This paper presents a multi-objective particle swarm optimization algorithm to efficiently solve the DLAN topology design problem. Fuzzy logic is incorporated in the PSO algorithm to handle the multi-objective nature of the problem. Specifically, a recently proposed fuzzy aggregation operator, namely the unified And-Or operator [30], is used to aggregate the objectives. The proposed fuzzy PSO (FPSO) algorithm is empirically evaluated through a preliminary sensitivity analysis of the PSO parameters. FPSO is also compared with fuzzy simulated annealing and fuzzy ant colony optimization algorithms. Results suggest that the fuzzy PSO is a suitable algorithm for solving the DLAN topology design problem.

Salman A. Khan

Computer Engineering Department, King Fahd University of Petroleum & Minerals, Dhahran 31261, Saudi Arabia E-mail: salmank@kfupm.edu.sa

Andries P. Engelbrecht

Department of Computer Science, University of Pretoria, Pretoria 0002, South Africa

E-mail: engel@cs.up.ac.za

Keywords Particle swarm optimization · Fuzzy logic · Multi-objective optimization · Unified And-Or operator · Network topology design

1 Introduction

Particle swarm optimization (PSO) is an optimization heuristic proposed by Kennedy and Eberhart [11][12]. The PSO algorithm is based on the sociological behavior associated with bird flocking [12]. The algorithm makes use of cognitive and social information among the individuals (particles) to find an optimal solution to an optimization problem. The algorithm can be used to solve a variety of complex optimization problems, in domains of both continuous and binary problems [13]. The popularity of PSO is growing with applications in diverse fields of engineering [53,55,58], medicine [1,27, 56,65], and social sciences [42].

Among many complex optimization problems, computer communication network topology design (CCNTD) is one problem that has received considerable attention during the past three decades. Significant research has been done to address many variants of the network topology design (NTD) problem, and a number of techniques have been proposed to find efficient solutions to these problems [10,15,20,21,29,35]. A CCNTD problem requires an objective or objectives to be optimized. Presence of constraints further amplifies the complexity of such problems. However, many of these approaches have not proven to be fully able to address the problem under consideration [18,29,34,46]. Local search techniques have been frequently used to optimize network topology design problems [18,29,34,46, 59]. However, these techniques generally do not perform well enough when multiple objectives need to be optimized and/or constraints are present [29,60–62]. Hence, iterative heuristics, such as evolutionary algorithms or swarm intelligence techniques seem to be appropriate approaches to solve the problem. Iterative heuristics have a tendency to escape a local optimum and can often find a global optimum solution in a reasonable amount of computational time.

The design of distributed local area networks (DLANs), such as campus networks or enterprize networks, is a complex multi-objective optimization problem. This problem requires simultaneous optimization of a number of design objectives, such as network delay, monetary cost, hop count between communicating pairs, and network reliability, subject to a set of design constraints. This optimization problem tends to have a solution space that grows exponentially with the problem size. Attempts have been made earlier to solve this specific problem with optimization techniques such as simulated evolution (SimE) [30], ant colony optimization (ACO) [32], and simulated annealing (SA) [31]. However, application of PSO to the DLAN topology design problem has not been reported in the literature. There are somewhat simpler versions of the DLAN topology

design problems which are NP-hard [15,17,20], and hence the DLAN topology design problem can be classified as an NP-hard problem.

Several methods for handling the multi-objective aspects have been reported in the literature. Some of the popular methods [39] are the weighted sum method, ε-constraint method, lexicographic ordering, goal programming, the goal attainment method, and fuzzy logic [64]. Fuzzy logic has received notable attention for multi-objective optimization (MOO) problems, with applications in various areas. Among various fuzzy operators, the ordered weighted averaging (OWA) operator, proposed by Yager [57], has been frequently used to aggregate multiple objectives into a single objective function. Recently, the Unified And-Or (UAO) operator [30], which exhibits mathematical properties similar to that of OWA, has been proposed to aggregate multiple objectives into a single objective function.

Most applications of PSO are for single-objective optimization problems. In the multi-objective domain, a number of PSO based approaches have been proposed (as discussed in Section 2.2), and there is still scope for further exploration. The development of a multi-objective PSO for the DLAN topology design problem is one step towards the assessment of the performance of PSO in multi-objective optimization, with application to a real-world design problem. Therefore, the focus of this paper is not to compare the variants and alterations proposed for PSO by many researchers, but rather the development of a fuzzy logic based multi-objective PSO, and a preliminary analysis of the fuzzy PSO with respect to the UAO operator. The rest of the paper is organized as follows: Section 2 provides the necessary background on PSO. Section 3 provides a short introduction to fuzzy logic and the unified And-Or operator. A brief description of the DLAN topology design problem is given in Section 4. Section 5 proposes a fuzzy PSO (FPSO) algorithm. Section 6 provides empirical results and a discussion of the performance of FPSO with respect to the UAO operator. A comparison with fuzzy simulated annealing (FSA) and fuzzy ant colony (FACO) algorithms is also given in Section 6. Conclusions are provided in Section 7.

2 Particle Swarm Optimization

In PSO, a population of potential solutions to the problem under consideration is used to explore the search space [44]. Each individual of the population is called a 'particle'. A particle has an adaptable velocity (step size), according to which the particle moves in the search space. Moreover, each particle has a memory, remembering the best position it has ever visited in the search space [14]. This best position is termed as the personal best, or *pbest*. The fitness value associated with the

pbest position is also recorded. Another "best value" that is tracked by the global version of the particle swarm optimizer is the overall best value, and the associated best location, obtained so far by any particle in the population. This location is called the *gbest* particle. Thus, a particle's movement is an aggregated 'acceleration' towards its best previously visited position (the cognitive component) and towards the best individual (the social component) of a topological neighborhood.

The particle swarm optimization algorithm consists of, at each time step, changing the velocity (accelerating) of each particle toward its *pbest* and *gbest* locations in the global version of the PSO. Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *gbest* locations.

Each particle in the swarm maintains the following information:

- $\mathbf{x}_i$: the *current position* of the particle;
- $\mathbf{v}_i$: the *current velocity* of the particle;
- $\mathbf{y}_i$: the *personal best position* of the particle; and
- $\hat{\mathbf{y}}_i$: the *neighborhood best position* of the particle.

The velocity update step is specified separately for each dimension, $j \in 1...N$, where $v_{i,j}$ represents the j^{th} dimension of the velocity vector associated with the i^{th} particle. The velocity of particle i is updated using

$$v_{i,j}(t+1) = wv_{i,j}(t) + c_1 r_{1,j}(t)[y_{i,j}(t) - x_{i,j}(t)] + c_2 r_{2,j}(t)[\hat{y}_j(t) - x_{i,j}(t)] \quad (1)$$

where w is the *inertia weight*, c_1 and c_2 are *acceleration coefficients*, and $r_{1,j}, r_{2,j} \sim U(0,1)$ are two independent random numbers sampled from a uniform distribution between 0 and 1. These random numbers induce a stochastic component in the search process.

The position $\mathbf{x}_i$ of a particle i is updated using

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \quad (2)$$

Figure 1 lists pseudo-code of the basic PSO. There are many main variations of this PSO algorithm [16]. Based on the neighborhood topology used, two early versions of PSO have been developed [44]: the global best (*gbest*) PSO, and the local best (*lbest*) PSO.

The rest of this section discusses PSO parameters and MOO approaches using PSO.

2.1 PSO Parameters

The standard PSO algorithm consists of several parameters that have an influence on the performance of the algorithm [13]. These include

- **Dimensionality of the particles**

Algorithm PSO();

For each particle $i \in 1, ..., s$ **do**

Randomly initialize $\mathbf{x}_i$

Initialize $\mathbf{v}_i$ to zero

Set $\mathbf{y}_i = \mathbf{x}_i$

end For

Repeat

For each particle $i \in 1, ..., s$ **do**

Evaluate the fitness of particle i

Update $\mathbf{y}_i$

Update $\hat{\mathbf{y}}_i$

For each dimension $j \in 1, ..., N$ **do**

Apply velocity update using Equation (1)

end For

Apply position update using Equation (2)

end For

Until some convergence criterion is satisfied

end Algorithm

Fig. 1 Pseudo-code of the basic particle swarm optimization algorithm

Usually, dimensionality is considered an important parameter in determining the hardness of a problem. PSO has been shown to perform very well on a wide variety of hard, high-dimensional benchmark functions such as the De Jong suite and other hard problems including Schaffer's f6, Griewank, Ackley, Rastrigin, and Rosenbrock functions [3, 12, 50]. Angeline [13] found that PSO actually performs relatively better on higher-dimensional versions of some test functions than on versions of the same functions in fewer dimensions.

– **Number of particles (i.e. swarm size)**

Swarm size is another important factor in PSO. Increasing population size generally causes an increase in computational complexity per iteration, but favors higher diversity, and therefore, may take less iterations to converge [13]. Generally, there is an inverse relationship between the size of the population and the number of iterations required to find the optimum of an objective function [13].

– **Inertia weight, w**

The inertia weight w is a modification to the standard PSO, proposed by Shi and Eberhart [50], to control the impact of the previous history of velocities on the current velocity, thus influencing the trade-off between global (wide-ranging) exploration and local (nearby) exploitation abilities of the particles. A larger value of w facilitates exploration (searching new areas), thus increasing diversity. A smaller value of w tends to facilitate local exploitation to fine-tune the current search area.

– **Acceleration coefficients $\mathbf{c}_1$ and $\mathbf{c}_2$**

The acceleration coefficients, c_1 and c_2, associated with the cognitive and social components play an important role in the convergence ability of the PSO. Varying these parameters has the effect of varying

the strength of the pull towards the two bests (i.e. personal best and neighborhood best). Values of $c_1 = c_2 = 0$ mean that both the cognitive and social components are absent, and particles keep moving at their current speed until they hit a boundary of the search space (assuming no inertia) [16]. With $c_1 > 0$ and $c_2 = 0$, each particle searches for the best position in its neighborhood, and replaces the current best position if the new position is better [16]. However, with $c_2 > 0$ and $c_1 = 0$, the entire swarm is attracted to a single point, $\hat{\mathbf{y}}$. Furthermore, having $c_1 >> c_2$ causes each particle to be attracted to its own personal best position to a very high extent, resulting in excessive wandering. On the other hand, $c_2 >> c_1$ results in particles being more strongly attracted to the global best position, thus causing particles to rush prematurely towards optima [16]. Van den Bergh [54] showed that the relation between acceleration coefficients and inertia weight should satisfy the following equation to have guaranteed convergence:

$$\frac{c_1 + c_2}{2} - 1 < w < 1 \tag{3}$$

– **Velocity clamping, $\mathbf{V_{max}}$**

Since there was no actual mechanism for controlling the velocity of a particle, it was necessary to impose a maximum value, V_{max}, on it [14]. V_{max} restricts the step size, i.e. the amount by which velocity is updated. This upper limit on step sizes prevents individuals from moving too rapidly from one region of the problem space to another, overshooting good regions of the search space. V_{max} proved to be crucial, because large values could result in particles moving past good solutions, while small values could result in insufficient exploration of the search space due to too small step sizes. The value assigned to V_{max} is not arbitrary, and should be optimized for each problem. It is recommended to set V_{max} to a value that is determined by the domain of the variables [13].

2.2 PSO and Multi-objective Optimization

PSO was also adapted to solve MOO problems. Reyes-Sierra and Coello-Coello [49] provided a detailed classification of current MOO approaches for PSO, as discussed below:

1. **Aggregating approaches**

   This category considers approaches that "aggregate" all the objectives of the problem into a single one. In other words, the multi-objective problem is converted into a linear combination of the sub-objectives. PSO aggregation approaches were proposed by Par-

sopoulos and Vrahatis [45] and Baumgartner *et al.* [6].

2. **Lexicographic ordering**

   Lexicographic ordering [39] has also been applied to multi-objective PSO [25,26]. Lexicographic ordering [39] ranks the objectives in order of importance. The domain expert (modeler) assigns the importance to objectives. The optimum solution, $\mathbf{x}^*$, is then obtained by optimizing the objective functions. The most important objective is optimized first, after which the other objectives are optimized according to the assigned order of their importance.

3. **Sub-Swarm approaches**

   Sub-swarm approaches use one swarm for each objective. That is, each swarm optimizes one of the sub-objectives. An information exchange mechanism is used to balance the trade-offs among the different solutions generated for the objectives that were separately optimized [7,43,49].

4. **Pareto-based approaches**

   Pareto-based approaches involve "leader selection" techniques based on Pareto dominance. In MOO PSO, the leaders are the personal best positions (local leaders) and neighborhood best positions (global leaders). The basic idea is to select leaders to the particles that are non-dominated with respect to the rest of the swarm [5,9,19,24,40,41,47,49].

3 Fuzzy Logic and Multi-objective Optimization

The theory of fuzzy logic [64] is based on a multi-valued logic wherein a statement could be partly true and partly false at the same time. A fuzzy logic approach differs from binary logic, in that binary logic allows a statement to be either false or true. The binary logic approach mathematically maps to a crisp set, X, where each element $x \in X$ can either belong to a set or not. In contrast to binary logic, fuzzy logic establishes an approximate truth value of a proposition based on linguistic variables and inference rules. Thus, in fuzzy sets, an element may partially belong to a set. This partial belonging is expressed by a membership function, μ, in the range [0,1], where μ is used to determine the degree of membership to the fuzzy set.

Like crisp sets, set operations such as intersection, union, and the complement, are also defined on fuzzy sets. Zadeh [63] first suggested to implement the 'AND' and the 'OR' as 'min' and 'max' operations. However, in certain multi-objective applications, Zadeh's 'AND' and 'OR' operators appeared to be quite rigid [36]. Over the years, many refinements have been proposed to Zadeh's operators. Some of these refinements resulted as probability operators [36], bounded operators [36], Einstein's operators [36], Hamacher's opera-

tors [22], Yager's ordered weighted average (OWA) operator [57], and the unified And-Or (UAO) operator [30].

3.1 The Unified And-OR Operator

The Unified And-Or (UAO) operator [30] is a modification of Yager's OWA operator. Khan and Engelbrecht [30] showed that the UAO operator also satisfies the monotonicity, symmetry, and idempotency conditions, as does the OWA operator. One important characteristic of the UAO operator is that it allows easy adjustment of the degree of "anding" and "oring" embedded in the aggregation, just like the OWA operator. The main difference between the two operators is that the OWA has two separate equations to represent the AND and the OR functions. However, UAO uses a single equation, yet the operator is capable of behaving either as the OWA-AND or the OWA-OR operator. The behavior is controlled by a variable $\nu \geq 0$, whose value decides whether the function will behave as AND or OR. The operator is defined as

$$f(a,b) = \frac{ab + \nu \max\{a,b\}}{\nu + \max\{a,b\}} = \begin{cases} \mathrm{I}_\star = \mu_{A \cup B}(x) \text{ if } \nu > 1 \\ \mathrm{I}^* = \mu_{A \cap B}(x) \text{ if } \nu < 1 \end{cases} \tag{4}$$

where a represents the membership value of μ_A (i.e. $a = \mu_A$), b represents the membership value of μ_B (i.e. $b = \mu_B$), and $f(a,b)$ represents the value of the overall objective function (i.e. $f(a,b) = \mu_{AB}$). I^* represents the AND operation using the UAO operator, and $I_\star$ denotes the OR operation using the UAO operator. For further details of the UAO operator, the interested reader is referred to Khan and Engelbrecht [30].

4 DLAN Topology Design Problem

The DLAN topology design problem requires finding a quality feasible tree topology under a given set of design objectives and constraints. This tree topology will interconnect all nodes (LANs) in the network, thus forming a backbone topology of a DLAN. The term "feasible topology" refers to a solution that satisfies all design principles and constraints. The term "quality topology" refers to a solution that optimizes the design objectives. In this paper, the quality of a topology is evaluated based on four design objectives: monetary cost, average network delay per packet (network latency), maximum number of hops between any source-destination pair, and network reliability. The search targets feasible topologies which minimizes cost, average network delay, and maximum hops, while maximizing network reliability.

4.1 Nomenclature

The following symbols are used throughout the paper:

n	number of nodes (i.e. LANs).
d	total number of networking devices in the network, where nodes are connected to networking devices.
T	$n \times n$ topology matrix where, $t_{ij} = 1$ if LANs i and j are connected and $t_{ij} = 0$ otherwise.
λ_i	traffic in bits per second (bps) on link i.
$\lambda_{max,i}$	capacity in bps of link i.
L	number of links of the proposed tree topology.
D_{nd}	delay due to network devices.
b_{ij}	delay per packet.
ω	average packet size in bits.
B_{ij}	delay per bit due to the network device feeding the link connecting LANs i and j, equal to $b_{i,j}/\omega$.
p_i	maximum number of nodes that can be connected to node i.
γ_{ij}	external traffic in bps between nodes i and j.
γ	overall external traffic in bps.
R_s	reliability of the network.
R_i	reliability of a link.

4.2 Design Objectives

As mentioned above, four conflicting design objectives are considered. These objectives are discussed below.

4.2.1 Monetary cost

The aim is to find a topology with low cost, while satisfying the design constraints (refer to Section 4.3). The only factor that affects the monetary cost is the cost of cables, as the number of network devices would be the same in any topology. Cost is expressed as

$$cost = \ l \times c_{cable} \tag{5}$$

where l represents the total length of cable, and c_{cable} represents the cost per unit of the cable used.

4.2.2 Average Network Delay

The second objective is to minimize the average network delay incurred on a packet during transmission from a source node to a destination node.

Estimation of the average network delay is done using the aggregate behavior of a link and network device. This behavior is modelled by an M/M/1 queue [15]. If a link connects local sites i and j, then the delay per bit due to the network device feeding this link is $B_{i,j} = b_{i,j}/\omega$. If γ_{ij} is the total traffic through the network device between local sites i and j, then the average packet delay due to all network devices is:

$$D_{nd} = \frac{1}{\gamma} \sum_{i=1}^{d} \sum_{j=1}^{d} \gamma_{ij} B_{ij} \tag{6}$$

where γ is the sum of all γ_{ij}. The total average network delay is the summation of delays of links and network devices, given by the following equation [15],

$$D = \frac{1}{\gamma} \sum_{i=1}^{L} \frac{\lambda_i}{\lambda_{max,i} - \lambda_i} + \frac{1}{\gamma} \sum_{i=1}^{d} \sum_{j=1}^{d} \gamma_{ij} B_{ij} \tag{7}$$

4.2.3 Maximum number of hops between any source-destination pair

The maximum number of hops between any source-destination pair is to be minimized. A hop is counted as the packet crosses a network device. The reason for taking number of hops as an optimization objective is due to the restrictions imposed by the routing information protocol (RIP). RIP uses hop count to measure the distance between the source and a destination node. RIP implements a limit on the number of hops encountered in the path from a source to a destination to prevent routing loops from continuing indefinitely [51]. The maximum number of hops allowed in a path is 15. If the hop count exceeds this number, then the destination is considered unreachable [51].

4.2.4 Network reliability

Network reliability is to be maximized. Network reliability can be defined as the probability of occurrence of an event in which each node communicates with every other node in the network [2]. In our case, the topology is a tree. Thus, the reliability of such a topology is the product of the reliabilities of all links present in that particular topology [28, 4]. Mathematically,

$$R_s = \prod_{i=1}^{L} R_i \tag{8}$$

4.3 Constraints

Three types of constraints are considered in this paper.

1. The objective of the first type of constraint is to ensure that the maximum number of nodes attached to a network device i must not exceed the capacity p_i of that device. That is,

$$\sum_{j=1}^{n} t_{ij} < p_i \qquad i = 1, 2, ..., n \qquad \forall i \neq j \tag{9}$$

2. The second type of constraint is dictated by bandwidth limitation of the links. A good network will employ "reasonably" utilized links. High utilization levels cause delays, congestion, and packet loss. Thus the traffic flow on any link i must be limited by a threshold value, $\lambda_{max,i}$:

$$\lambda_i < \lambda_{max,i} \quad i = 1, 2, ..., s \tag{10}$$

where s is the total number of links present in the current topology.

3. The third set of constraints are specified by the designer, and is used to enforce certain design guidelines and principles, for example,
   (a) Certain nodes must be leaf/terminal nodes. For example, hubs should generally be placed as leaf nodes.
   (b) Certain nodes must be interior nodes of the tree, for example, nodes designated as switches or routers.
   (c) Certain nodes cannot be directly connected to the backbone. For example, hubs (and in some cases, switches) should not be directly connected to the backbone (i.e. the root node).

4.4 Fuzzy logic approach for the DLAN topology design problem

As discussed earlier, the DLAN topology design is a complex optimization problem with a huge search space. The design objectives are conflicting. The complexity of the problem is further amplified by the presence of constraints. Khan and Engelbrecht [30] discussed in detail that fuzzy logic is a suitable approach to address the above complex problem. Note that the four design objectives (i.e. cost, delay, hops, and reliability) have different units and scales. For example, cost is in dollars and can be in millions, delay is in milliseconds, hops is an integer between 1 and 14, while reliability is a fraction between 0 and 1. In order to aggregate these four objectives, the objectives need to be normalized to the same scale, and that is done through membership functions in fuzzy logic. However, other methods, such as lexicographic ordering or sub-swarm approaches, do not necessarily require conversion of these different objectives into a same (normalized) scale and units; the multi-objective optimization can still be performed without changing the scale and units of these objectives. These other approaches can also be used with a multi-objective PSO to solve the DLAN topology design problem.

In fuzzy logic, the four design objectives of the DLAN topology design problem can be combined using the following fuzzy rule:

> Rule 1: **IF** a solution X has *low cost* AND *low delay* AND *low hops* AND *high reliability*
> **THEN** it is a *good topology.*

The expressions “low cost”, “low delay”, “low hops”, “high reliability”, and “good topology” are linguistic values, each defining a fuzzy subset of solutions. For example, “high reliability” is the fuzzy subset of topologies of high reliabilities. Each fuzzy subset is defined by a membership function, μ. The membership function returns a value in the interval [0,1] which describes

the degree of satisfaction with the particular objective criterion. To find the membership functions of the individual objectives, we proceed as follows.

The membership function of cost is defined by first determining two extreme values - the minimum and maximum costs. These values could be found mathematically or from prior knowledge. The membership value for cost, μ_c, is then computed as

$$\mu_c(x) = \begin{cases} 1 & \text{if } Cost \leq MinC \\ \frac{MaxC-Cost}{MaxC-MinC} & \text{if } MinC < Cost \leq MaxC \\ 0 & \text{if } Cost > MaxC \end{cases} \tag{11}$$

where the term *Cost* represents the cost of the solution, 'MinC' represents the lower limit of cost and 'MaxC' represents the maximum limit of cost. The membership function of delay, μ_d, can be defined in a similar way. The two extreme values of delay are 'MinD' and 'MaxD' for minimum and maximum delay respectively. The membership value of delay is determined as

$$\mu_d(x) = \begin{cases} 1 & \text{if } Delay \leq MinD \\ \frac{MaxD-Delay}{MaxD-MinD} & \text{if } MinD < Delay \leq MaxD \\ 0 & \text{if } Delay > MaxD \end{cases} \tag{12}$$

where the term *Delay* represents the average delay of the solution. Similarly, the membership function for number of hops, μ_h, with 'MinH' and 'MaxH' as the lower and upper limits respectively, is determined as

$$\mu_h(x) = \begin{cases} 1 & \text{if } Hops \leq MinH \\ \frac{MaxH-Hops}{MaxH-MinH} & \text{if } MinH < Hops \leq MaxH \\ 0 & \text{if } Hops > MaxH \end{cases} \tag{13}$$

Finally, the membership function for reliability, μ_r, can be determined by finding the maximum (MaxR) and the minimum (MinR) bounds for the reliability. The membership value for reliability is determined as

$$\mu_r(x) = \begin{cases} 1 & \text{if } Rel \geq MaxR \\ \frac{MaxR-Rel}{MaxR-MinR} & \text{if } MinR < Rel \leq MaxR \\ 0 & \text{if } Rel < MinR \end{cases} \tag{14}$$

Once the individual membership values are found using the above functions, Rule 1 can be mathematically represented using the UAO operator as:

$$\mu(x)^{I_\star} = \frac{\prod_{i=1}^{4} \mu_i(x) + \nu \max\{\mu_1(x), \mu_2(x), \mu_3(x), \mu_4(x)\}}{\nu + \max\{\mu_1(x), \mu_2(x), \mu_3(x), \mu_4(x)\}} \tag{15}$$

In Equation (15), $\mu(x)^{I\star}$ represents the membership value of solution x in the fuzzy set *good topology* using the UAO operator. Also, μ_i for $i = \{1,2,3,4\}$ represents the membership values of solution x in the fuzzy sets *low cost*, *low delay*, *low hops*, and *high reliability* respectively. The solution which results in the maximum value for Equation (15) is reported as the best solution found. However, note that the algorithm generates Pareto optimal solutions (i.e., all best solutions on the Pareto front having equal membership values), and any one of these solutions can be taken as the best solution.

A detailed description on formation of membership functions for individual objectives can be found in [30, 31].

5 Fuzzy Particle Swarm Optimization Algorithm

The fuzzy PSO (FPSO) maintains a population of particles. Each particle is responsible for generating a feasible network topology. A particle in PSO progresses iteration by iteration, learning from its own history, and it also inherits characteristics from other particles generating high-quality solutions. This is done while simultaneously considering the design objectives and constraints. In FPSO, a particle incrementally improves an already existing solution. This improvement is done by replacing low-quality links with high-quality ones. The guidance in selection of links is provided by three parameters: the particle's current position, its own best position so far, and the best position in relation to the particle's neighborhood. Each step of the proposed FPSO is discussed next.

5.1 Particle Position and Velocity Representation

For the original PSO, particle position as well as velocity representation were in the real number domain, that is, all $x_{ij} \in \Re$, and all $v_{ij} \in \Re$. However, the DLAN topology design problem has discrete-valued variables. Therefore, the representation of particle positions and velocities need to change. This paper uses a set representation. This representation scheme is described below.

A position will be the set,

$$\mathbf{X}_i(t) = \{l_1, l_2, ..., l_q, ..., l_L\}$$

where l_q is a link between any two nodes a and b in the network, and L is a constant that represents the number of links in the solution, with $L = n - 1$, i.e. $|\mathbf{X}_i(t)| = L$. The velocity of particle i is represented as

$$\mathbf{V}_i(t) = \{l_q \Leftrightarrow l_q{}'\}$$

where link l_q is removed and replaced with link $l_q{}'$, and $|\mathbf{V}_i(t)|$ gives the total number of changes to particle i.

Example 1: Consider a simple network of 6 nodes as given in Figure 2. Note that $L = 5$ for any configuration

of this network. The topology in this figure represents a possible configuration at time t, and thus represents a solution (i.e. particle). According to the network configuration in Figure 2, the current solution is given as

$$\mathbf{X}_i(t)=\{(1,2),\ (1,3),\ (3,5),\ (4,5),\ (4,6)\}$$

That is, there are links between nodes (1,2), (1,3), (3,5), (4,5) and (4,6). This $\mathbf{X}_i(t)$ is also used in Examples 2 and 3 below.

Also assume that at time t, $\mathbf{V}_i(t) = \{(2,4) \Leftrightarrow (1,2),\ (3,4) \Leftrightarrow (3,5), (5,6) \Leftrightarrow (4,6)\}$ where the symbol "$\Leftrightarrow$" represents an exchange of links. That is, the current solution $\mathbf{X}_i(t)$ was obtained when link (2,4) was removed and replaced with (1,2), then (3,4) was removed and replaced with (3,5), and then (5,6) was removed and replaced with (4,6). This $\mathbf{V}_i(t)$ is also used in Examples 2 and 3 below.

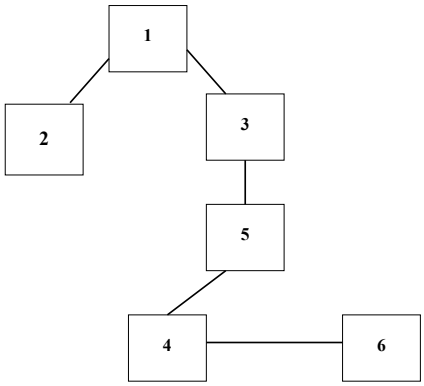

Fig. 2 Network topology for PSO example

5.2 Velocity Update

The velocity of particle i is updated using

$$\mathbf{V}_i(t+1) = w \otimes \mathbf{V}_i(t) \oplus c_1 r_1(t) \otimes [\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)] \oplus$$

$$c_2 r_2(t) \otimes [\mathbf{P}_g(t) \oslash \mathbf{X}_i(t)] \quad (16)$$

where $\mathbf{P}_i(t)$ represents the particle's own best position, and $\mathbf{P}_g(t)$ represents the global best position.

In Equation (16), the operator $\otimes$ is implemented as follows: the number of elements to be selected are determined as $\lfloor \mathrm{w} \times |\mathbf{V}_i(t)| \rfloor$. Then, the result will be the above number of elements randomly selected from $\mathbf{V}_i(t)$. The same approach is applicable to other terms where the operator $\otimes$ is used.

The operator $\oslash$ is implemented as the 'exchange' operator. For example, the links in $\mathbf{X}_i(t)$ are replaced with the links in $\mathbf{P}_i(t)$.

The term $c_1 r_1(t) \otimes [\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)]$ is implemented by multiplying c_1 and $r_1(t)$ with the size of the set $\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)$ and taking the floor, i.e.

$$c_1 r_1(t) \otimes [\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)] = \lfloor c_1 r_1 \times |\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)| \rfloor \quad (17)$$

where $|\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)|$ represents the cardinality of the set. The result of Equation (17) indicates the number of elements that are randomly selected from the set

$\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)$; $c_2 r_2(t) \otimes [\mathbf{P}_g(t) \oslash \mathbf{X}_i(t)]$ has the same meaning.

The operator $\oplus$ implements the set addition (union) operator, i.e. the elements in any two sets are combined in a new set using the set addition operator. Furthermore, V_{max} is used to limit the number of elements selected from a set.

Example 2: Continuing with Example 1, assume the following parameter values:

w = 0.5

$V_{max} = 2$

$c_1 = c_2 = 0.5$

$r_1 = 0.52$ (randomly generated)

$r_2 = 0.75$ (randomly generated)

Assume that the best goodness so far for particle i was generated by position,

$$\mathbf{P}_i(t) = \{(1,2),(1,4),(2,3),(2,5),(2,6)\}$$

Also assume that the best solution so far generated by the entire swarm was achieved by the following global best solution:

$$\mathbf{P}_g(t) = \{(1,2),(1,3),(1,4),(1,5),(1,6)\}$$

The inertia weight, w, determines the number of moves that will be randomly selected from $\mathbf{V}_i(t)$ (mentioned in Example 1 above). Since $w = 0.5$, and $|\mathbf{V}_i(t)| = 3$, the number of moves selected is $0.5 \times |\mathbf{V}_i(t)| = 1.5$. Since fractional moves are not possible, the value is truncated to 1.

Now, $0.5 \times \mathbf{V}_i(t) = \{(2,4) \Leftrightarrow (1,2)\}$. Note that $(3,4) \Leftrightarrow (3,5)$ OR $(5,6) \Leftrightarrow (4,6)$ is also possible; any one move of these three moves can be randomly chosen.

The difference between the particle's current position and its own best position, $\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)$, is calculated by replacing each link in $\mathbf{X}_i(t)$ with the link in the corresponding position in $\mathbf{P}_i(t)$ as

$\mathbf{P}_i(t) \oslash \mathbf{X}_i(t) = \{(1,2) \Leftrightarrow (1,2),(1,3) \Leftrightarrow (1,4),(3,5) \Leftrightarrow (2,3),(4,5) \Leftrightarrow (2,5),(4,6) \Leftrightarrow (2,6)\}$

Therefore, $c_1 \times r_1 \otimes (\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)) = 0.5 \times 0.52 \times |\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)|$. Since the cardinality of $\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)$ is 4 (i.e. there are four exchanges in the set, as $(1,2) \Leftrightarrow (1,2)$ is not considered an exchange), this implies that $0.5 \times 0.52 \otimes |\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)| = 1.04 = 1$. This means that any one of the four elements in $\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)$ can be randomly chosen. So, assume that $c_1 \times r_1 \otimes (\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)) = \{(4,6) \Leftrightarrow (2,6)\}$.

Similarly:

$\mathbf{P}_g(t) \oslash \mathbf{X}_i(t) = \{(1,2) \Leftrightarrow (1,2),(1,3) \Leftrightarrow (1,3),(3,5) \Leftrightarrow (1,4),(4,5) \Leftrightarrow (1,5),(4,6) \Leftrightarrow (1,6)\}$

The cardinality of the above set is 3, since $(1,2) \Leftrightarrow (1,2)$ and $(1,3) \Leftrightarrow (1,3)$ are not considered exchanges. So, $0.5 \times 0.75 \otimes (\mathbf{P}_g(t) \oslash \mathbf{X}_i(t)) = 0.5 \times 0.75 \times 3 = 1.12 = 1$ move. Assume $\{(4,5) \Leftrightarrow (1,5)\}$ is randomly chosen,

although any combination consisting of a single move from $\mathbf{P}_g(t) \oslash \mathbf{X}_i(t)$ can be randomly chosen.

Substituting the above calculations in Equation (16) gives $\mathbf{V}_i(t+1)$ containing three elements as

$\mathbf{V}_i(t+1) = \{(2,4) \Leftrightarrow (1,2), (4,6) \Leftrightarrow (2,6), (4,5) \Leftrightarrow (1,5)\}$

Since $V_{max} = 2$, only two moves (i.e. exchanges) from $\mathbf{V}_i(t+1)$ can be randomly chosen. Assume that $(2,4) \Leftrightarrow (1,2)$ and $(4,6) \Leftrightarrow (2,6)$ are chosen.

Hence,

$$\mathbf{V}_i(t+1) = \{(2,4) \Leftrightarrow (1,2), (4,6) \Leftrightarrow (2,6)\}$$

5.3 Particle Position Update

The position $\mathbf{X}_i(t)$ of a particle i is updated using

$$\mathbf{X}_i(t+1) = \mathbf{X}_i(t) \boxplus \mathbf{V}_i(t+1) \quad (18)$$

where $\boxplus$ is a special operator that updates the links in $\mathbf{X}_i(t)$ on the basis of link exchanges in $\mathbf{V}_i(t+1)$, to get the new position $\mathbf{X}_i(t+1)$.

Example 3: Continuing with Example 2,

$\mathbf{X}_i(t+1) = \mathbf{X}_i(t) \boxplus \mathbf{V}_i(t+1) = \{(1,2), (1,3), (3,5), (4,5), (4,6)\} \boxplus \{(2,4) \Leftrightarrow (1,2), (4,6) \Leftrightarrow (2,6)\} = \{(1,2), (1,3), (3,5), (4,5), (2,6)\}$

Notice that since the link (2,4) was not present in $\mathbf{X}_i(t)$, the exchange $(2,4) \Leftrightarrow (1,2)$ could not be performed. Therefore, in the new solution, the links (1,2), (1,3), (3,5), and (4,5) have been brought from the solution $\mathbf{X}_i(t)$, while the new link, i.e. (2,6), was introduced, replacing the link (4,6), as specified by the replacement in $\mathbf{V}_i(t+1)$.

5.4 Fitness Evaluation

The fitness (goodness) of a solution is evaluated using Equation (15), as discussed in Section 4. During this evaluation process, the three constraints are also checked through a subroutine. After each move is performed, the subroutine checks whether the move has resulted in any violation of the constraints. If so, the move is reversed, and another move is performed. This is done until the allowed number of moves are done.

5.5 Initialization

Since PSO is a population-based algorithm, the initialization process consists of generating a set of candidate solutions. These initial solutions are generated randomly, and constraints are checked at each step to ensure feasible initial solutions. The goodness of each particle is then calculated using Equation (15). Algorithm parameters such as inertia weight, velocity clamping, and acceleration constants are also initialized.

5.6 Particle Activity

For the FPSO, the *gbest* model has been used (although the *lbest* model could be applied as well). Once the initial set of solutions is generated, the global best particle is chosen on the basis of the goodness value calculated in the initialization phase. Note that, at this stage, each particle's current position is its best position. In the following iterations, each particle updates its position based on information provided by the particle's immediate previous position and by the alterations (moves) performed on the particle through the velocity update vector, as explained in Section 5.3. The velocity of a particle is updated on the basis of moves performed on the particle in its immediate previous position, the particle's own best position so far, and the overall best position achieved by any particle in the swarm in any iteration, as described in Section 5.2. Moreover, to avoid premature convergence, the global best particle is updated regularly, i.e. as soon as a particle's overall goodness becomes higher than the overall goodness of the global best particle, that new particle is selected as the global best particle, and the search process continues. If no updating is done, then the algorithm will very quickly converge on a solution that might not even be a local minimum.

A 'move' in FPSO includes removing a link and introducing a new link such that the tree is maintained (refer to Example 1 in Section 5.1). Then, the constraints are checked to evaluate the feasibility of the performed move. However, the notion of moves in FPSO depends on three factors, namely: moves performed in the immediate previous position of the particle, the structure of the particle's own best position, and the structure of the global best particle. For all these factors, the number of moves performed to get the new position of the particle is governed by parameters such as acceleration coefficients, inertia weight, and velocity clamping. Values of these parameters decide how many moves are required to get the new position of a particle.

6 Results and Discussion

The fuzzy PSO was applied to the five test cases used in [30–32]. These test cases were named *n15*, *n25*, *n33*, *n40*, and *n50*, where the numerals in the test cases reflect the number of nodes (local sites) in the respective test case. These test cases represent randomly generated networks. Traffic generated by each local site for these test cases was collected from real sites, and costs of the network cables were collected from vendors. Other characteristics, such as the number of ports on a network device, its type, etc. are listed in Table 1. Table 2 summarizes the characteristics of these test cases.

Table 1 Network characteristics assumed for experiments.

Parameter	Characteristic
Cost of fiber optic cable	$ 5 per meter
Delay per bit due to networking device	250μsec.
Maximum traffic on a link allowed	60
Average packet size	500 bytes
Type of networking device	Router, switch, or hub
Number of ports on a networking device	4, 8, or 12

Table 2 Characteristics of test cases used in experiments. MinC is in US$, MinD is in milliseconds, and traffic is in Mbps.

Test Case	# of Local Sites	MinC	MinD	MaxR	Traffic
n15	15	4640	2.143	0.8687	24.63
n25	25	5120	2.151	0.7857	74.12
n33	33	8158	2.154	0.7250	117.81
n40	40	9646	2.088	0.6757	144.76
n50	50	11616	2.900	0.6111	164.12

The performance of the algorithm was evaluated with respect to a number of parameters. These parameters are the swarm size, acceleration constants, inertia weight w, and velocity clamping V_{max}. The parameter values used in the experiments are given in Table 3. In these experiments, each instance of the algorithm was run for 100 iterations. Thirty independent runs were executed for each parameter setup, and the average of best solutions found in each run was reported, with the associated standard deviation. The following default values were used for experiments, unless otherwise specified: number of particles = 20, V_{max} = 5, w = 0.72, and $c_1 = c_2 = 0.5$.

Table 3 Parameter settings for fuzzy PSO used in experiments.

Parameter	Values
Number of particles	5, 10, 15, 20, 25, 30
V_{max}	5 10% size of test case 20% size of test case
w	0.72 0.95 0.4
c_1, c_2	0.5 and 0.5 1.49 and 1.49 2.0 and 2.0

6.1 Effect of Swarm size

The effect of swarm size was investigated with different number of particles as given in Table 3. Other parameters were kept at the default values given above. Tables 4 to 8 reflect the effect of the number of particles on solution quality. Column 1 lists the test case, column 2 gives the overall goodness obtained using the UAO operator, and column 3 lists the percentage difference between the average performance of the corresponding number of particles and the best goodness (in boldface). For example, in Table 4, the best overall goodness (in

boldface) using UAO is obtained with 30 particles. The overall goodness with different numbers of particles is then compared with the best overall goodness, and the percentage difference is listed. It is evident from Tables 4 to 8 that the best overall goodness was obtained when the number of particles was relatively high. More specifically, the best results were obtained in all cases with a swarm size of 30. A small deviation from this trend was the case *n33* where 25 particles produced the best overall goodness.

Table 4 Effect of swarm size on overall goodness for *n50* with UAO. Statistically significant difference is in italics.

No. of particles	Goodness (UAO)	% Difference
10	0.333 ±0.005	*0.89*
15	0.334 ±0.004	*0.60*
20	0.335 ±0.004	0.30
25	0.335 ±0.002	0.30
30	**0.336** ±0.003	NA

Table 5 Effect of swarm size on overall goodness for *n40* with UAO. Statistically significant difference is in italics.

No. of particles	Goodness (UAO)	% Difference
10	0.338 ±0.005	*3.70*
15	0.341 ±0.012	*2.85*
20	0.342 ±0.009	*2.56*
25	0.346 ±0.010	1.42
30	**0.351** ±0.012	NA

Table 6 Effect of swarm size on overall goodness for *n33* with UAO. Statistically significant difference is in italics.

No. of particles	Goodness (UAO)	% Difference
10	0.332 ±0.006	*2.06*
15	0.337 ±0.006	0.59
20	0.337 ±0.005	0.59
25	**0.339** ±0.005	NA
30	0.338 ±0.006	0.29

Table 7 Effect of swarm size on overall goodness for *n25* with UAO. Statistically significant difference is in italics.

No. of particles	Goodness (UAO)	% Difference
10	0.330 ±0.009	*2.65*
15	0.330 ±0.004	*2.65*
20	0.335 ±0.005	1.18
25	0.337 ±0.008	0.60
30	**0.339** ±0.008	NA

Table 8 Effect of swarm size on overall goodness for *n15* with UAO. Statistically significant difference is in italics.

No. of particles	Goodness (UAO)	% Difference
10	0.332 ±0.002	0.32
15	0.332 ±0.002	0.32
20	0.332 ±0.001	0.32
25	0.332 ±0.002	0.32
30	**0.333** ±0.003	NA

A t-test validation of percentage difference in Tables 4 to 8 was also performed, and the statistically significant differences are given in italics. An important observation in Tables 4 to 8 is that the lowest level

of overall goodness was obtained when the number of particles was lowest. More specifically, having 10 particles resulted in the worst solutions in all cases. The only exception to this trend was *n15*. In this instance, all particles from 10 to 25 resulted in the same overall goodness value. The improvement between the highest and the lowest overall goodness using the UAO operator is given in Table 9, which shows that the improvements were generally less than 4%, but all improvements (except for *n15*) were statistically significant, as validated by the t-test.

A graphical representation of the results in Tables 4 to 8 is given in Figure 4. This figure shows the effect on overall goodness when the number of particles are varied from 10 to 30. This figure further strengthens the observations, noted above, that in general, increasing the number of particles positively affects the quality of overall goodness of the solution. For example, in Figure 4(a), the overall goodness increased with an increase in the number of particles for case *n50*.

The above discussion and observations suggest that, in general, an increase in the number of particles increases diversity and reduces the possibility of getting trapped in local minima, thereby resulting in higher quality solutions. The statement can be further supported by the plots in Figure 3 as an example. The figure shows a typical diversity plot for 10 and 30 particles for *n50*. Diversity is essentially a measure of the average distance of each particle from the center of the mass, and is calculated at each iteration during the execution of the algorithm. As can be observed in the figure, both swarms (with 10 and 30 particles) do not begin converging immediately following initialization, but they maintain their diversity, and rather expand slightly. More specifically, the FPSO with 30 particles expands substantially compared to FPSO with 10 particles. For example, after the first ten iterations, the diversity of particles in the FPSO with 10 particles increases from 0.029 to 0.041, while in FPSO with 30 particles, swarm diversity increases from 0.036 to 0.063.

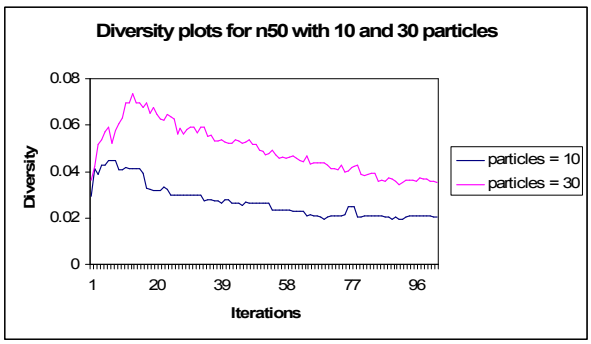

Fig. 3 Diversity plots for *n50* using 10 and 30 particles

6.2 Effect of Acceleration Coefficients

The effect of acceleration coefficients was investigated with different values of the coefficients as given in Table 3. Other parameters were kept at the defaults. Table 10 provides the results for the UAO operator with respect

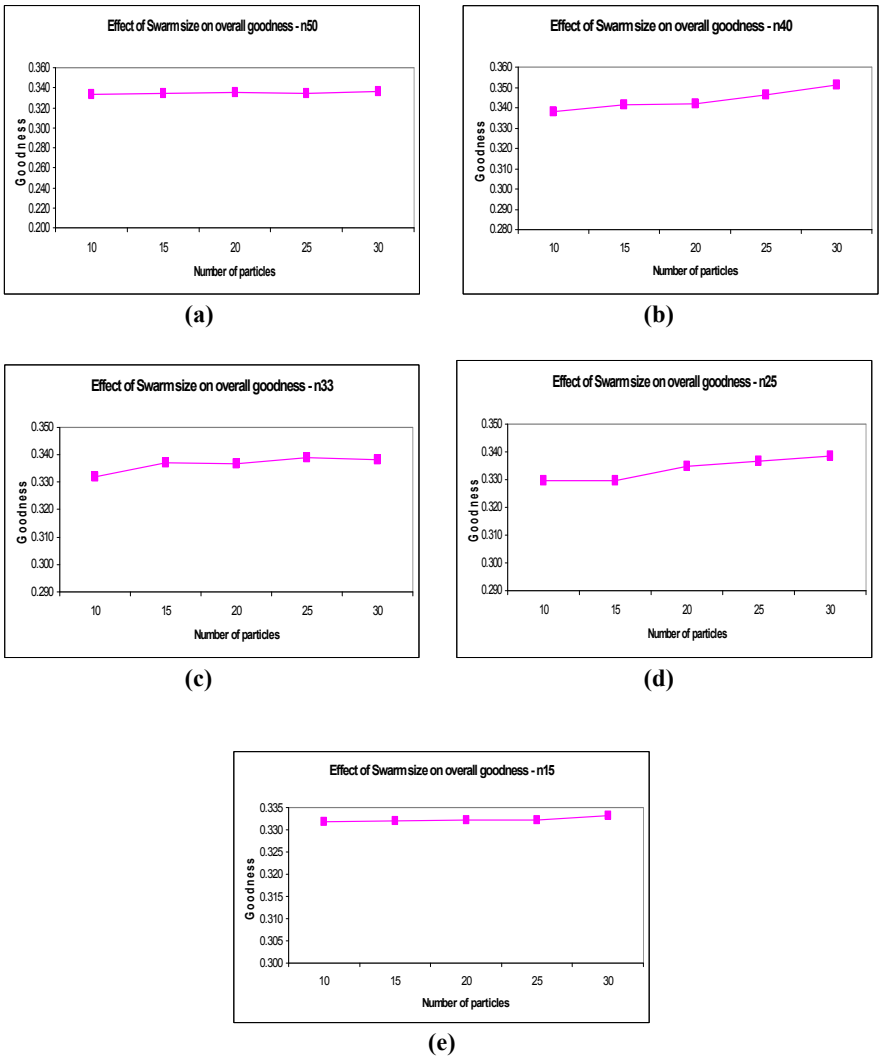

Fig. 4 Effect of swarm size on overall goodness for (a) n50 (b) n40 (c) n33 (d) n25 (e) n15

Table 9 Results for best and worst average overall goodness and their respective number of particles for UAO. Statistically significant improvement is in italics.

Case	Particles	Max. goodness	Particles	Min. goodness	% improvement
n15	30	0.333 ±0.003	10,15,20,25	0.332 ±0.002	0.32
n25	30	0.339 ±0.008	10	0.330 ±0.009	*2.69*
n33	25	0.339 ±0.005	10	0.332 ±0.006	*2.06*
n40	30	0.351 ±0.012	10	0.338 ±0.005	*3.90*
n50	30	0.336 ±0.003	10	0.333 ±0.005	*0.89*

to the three sets of acceleration coefficients. The values $c_1 = c_2 = 1.49$ (along with inertia weight = 0.72) were specifically chosen, since they are often used in the literature and they ensure convergence [52].

Table 10 shows that each set of acceleration coefficients produced results of almost the same quality when compared with the other set. It is observed in the table that the percentage improvements achieved by any set of c_1 and c_2 compared to another set was at most 1% in the majority of cases. An exception from this trend was the case of *n40*, where $c_1 = c_2 = 0.5$ achieved an improvement of 3.96% over $c_1 = c_2 = 1.49$, $c_1 = c_2 = 2.0$ achieved an improvement of 6.09% over $c_1 = c_2 = 1.49$, and $c_1 = c_2 = 2.0$ achieved an improvement of 2.22% over $c_1 = c_2 = 0.5$. A t-test validation showed that all improvements were insignificant, with the exception of *n40* when comparing $c_1 = c_2 = 1.49$ with other sets of c_1 and c_2. In general, the results indicate that the convergence of PSO is independent of the acceleration coefficients with respect to the values used.

6.3 Effect of Inertia Weight

The effect of the inertia weight, w, is empirically investigated in this section. Table 11 shows the results obtained for the fuzzy PSO with the UAO operator. The effect of w on performance was studied with three values, namely $w = 0.72$, $w = 0.95$, and $w = 0.4$. Other parameters were kept at the defaults.

Table 11 suggests that the difference between the overall goodness achieved by the three values of inertia weight was generally less than 1% for all test cases. The t-test showed that the improvements were insignificant. These observations suggest that the fuzzy PSO was insensitive to the inertia weight for the UAO operator with respect to the three values of w used.

6.4 Effect of Velocity Clamping

The effect of velocity clamping was also empirically studied. Table 12 shows the results obtained for fuzzy PSO with the UAO operator. The effect was studied with three values of velocity clamping, with one value fixed at $V_{max} = 5$ for all cases, while the other two were variable, proportional to the test case size. These variable values were $\lceil V_{max} = 10\% \rceil$ and $\lceil V_{max} = 20\% \rceil$ of the test case size. The inspiration for taking 10% and 20% the test case size comes from mutation rates in genetic algorithms. Note that both V_{max} in PSO and mutation rate in GA control the amount of perturbation of the solution, and therefore the functionality of both parameters is more or less the same. A number of studies [8,23,37,38] have used a mutation rate of up to 20% or more. Therefore, the basis of choosing a value for

Table 10 Effect of acceleration coefficients on the test cases, for UAO. % imp shows the improvement achieved by one set of values of c_1 and c_2 over the other set of values. Statistically significant improvement is in italics.

Case	$c_1 = c_2 = 0.5$ Goodness	$c_1 = c_2 = 1.49$ Goodness	$c_1 = c_2 = 2.0$ Goodness	% imp 0.5 vs 1.49	% imp 2.0 vs 1.49	% imp 2.0 vs 0.5
n15	0.332 ±0.001	0.333 ±0.002	0.332 ±0.001	-0.12	-0.12	0.0
n25	0.335 ±0.005	0.338 ±0.010	0.337 ±0.009	-1.03	-0.51	0.52
n33	0.337 ±0.005	0.338 ±0.006	0.336 ±0.004	-0.25	-0.64	-0.39
n40	0.342 ±0.009	0.328 ±0.060	0.350 ±0.011	*3.96*	*6.09*	2.22
n50	0.335 ±0.004	0.336 ±0.004	0.335 ±0.003	-0.13	-0.13	0.0

Table 11 Effect of inertia weight on the test cases, for UAO. % imp shows the improvement achieved by one value of w over the other value. Statistically significant improvement is in italics.

Case	$w = 0.72$ Goodness	$w = 0.95$ Goodness	$w = 0.4$ Goodness	% imp 0.72 vs 0.95	% imp 0.72 vs 0.4	% imp 0.95 vs 0.4
n15	0.332 ±0.001	0.332 ±0.001	0.332 ±0.002	0.0	0.0	0.0
n25	0.335 ±0.005	0.331 ±0.005	0.333 ±0.008	1.03	0.70	-0.33
n33	0.337 ±0.005	0.337 ±0.005	0.340 ±0.008	0.0	-0.81	-0.88
n40	0.342 ±0.009	0.344 ±0.011	0.345 ±0.011	-0.75	-0.91	-0.16
n50	0.335 ±0.004	0.335 ±0.003	0.335 ±0.004	0.0	0.0	0.0

V_{max} is this observation. Other PSO parameters were kept at the defaults.

Table 12 shows that velocity clamping had a very slight impact on the quality of overall goodness, with all values having less than 1.5% improvements. The t-test confirmed that all the improvements were statistically insignificant. Thus, the results in Table 12 suggest that velocity clamping did not have a significant effect on the quality of the overall goodness for the three values used for V_{max}.

6.5 Comparison with a Fuzzy Simulated Annealing Algorithm

The proposed FPSO was compared with a fuzzy simulated annealing (FSA) algorithm adapted for the DLAN topology design problem [31]. Simulated annealing [33]

Table 12 Effect of velocity clamping on the test cases, for UAO. % imp shows the improvement achieved by one value of V_{max} compared to the other value. NA = Not Applicable.

Case	$V_{max} = 5$ Goodness	$V_{max} = 10\%$ Goodness	$V_{max} = 20\%$ Goodness	% imp 5 vs 10%	% imp 5 vs 20%	% imp 10% vs 20%
n15	0.332 ±0.001	0.331 ±0.001	0.332 ±0.001	0.263	-0.009	-0.27
n25	0.335 ±0.005	0.332 ±0.007	The value of V_{max} is 5 here	0.843	NA	-0.84
n33	0.337 ±0.005	0.337 ±0.006	0.339 ±0.007	-0.024	-0.609	-0.58
n40	0.342 ±0.009	0.346 ±0.013	0.342 ±0.010	-1.207	-0.055	1.15
n50	0.335 ±0.004	The value of V_{max} is 5 here	0.335 ±0.005	NA	0.095	0.10

is a famous optimization algorithm and has been successfully applied to a number of complex optimization problems. In FSA for the DLAN topology design problem, the four design objectives were aggregated using the same approach as described in Section 4. However, the main difference between FSA and FPSO is that FSA maintains and perturbs a single solution throughout the execution of the algorithm, while FPSO perturbs a number of solutions during each iteration.

FSA has two main steps: initialization and the Metropolis procedure. The initialization phase randomly generates a feasible solution. This solution is then passed to the Metropolis procedure which perturbs the solution. If the overall goodness of the new solution (i.e., perturbed solution) is higher than the overall goodness of the current solution, then the new solution is definitely accepted. However, if the overall goodness of the new solution is less than the overall goodness of the current solution then the new solution is probabilistically accepted based on the *Metropolis criterion* given by $P(random < e^{-\Delta h/T})$, where $random$ is a random number in the range 0 to 1, T represents the *annealing temperature*, and Δh represents the difference in the overall goodness of the current solution and the new solution. However, if the new solution does not pass the Metropolis criterion, or if any of the constraints are violated, then the new solution is not accepted and the current solution is restored.

FSA has a number of control parameters that affect the performance of the algorithm. These parameters include the initial temperature, T_0, the cooling rate, α_{SA}, the constant, β_{SA}, and the length of Markov

Table 13 Comparison of FSA and FPSO for UAO. OG = overall goodness, par = number of particles, Time = average execution time in seconds, and % imp = percentage improvement achieved by FPSO compared to FSA. Statistically significant improvement is in italics.

Case	FSA			FPSO						% imp
	α	OG	Time	par	V_{max}	c_1, c_2	w	OG	Time	FSA vs. FPSO
n15	0.99	0.335 ±0.003	89.0	20	5	1.49	0.72	0.333 ±0.002	29.7	-0.60
n25	0.75	0.345 ±0.034	314.4	30	5	0.5	0.72	0.339 ±0.008	187.0	-1.74
n33	0.75	0.339 ±0.088	765.1	20	5	0.5	0.4	0.340 ±0.008	428.1	0.29
n40	0.85	0.374 ±0.066	1498.8	30	5	0.5	0.72	0.351 ±0.012	1659.2	*-6.15*
n50	0.85	0.350 ±0.053	4295.8	30	5	0.5	0.72	0.336 ±0.003	7074.9	*-4.00*

chain, M, which represents the time until the next parameter update. Inappropriate selection of values for these parameters may result in low quality solutions. A detailed study of the effects of these parameters on the FSA algorithm performance was done in [31]. The study used the following parameter values: the initial temperature was set at $T_0 = 1000$. For the cooling rate, α, values of 0.6, 0.75, 0.85, 0.95, and 0.99 were considered. The length of the Markov chain was set at M =10. The annealing constant was set at $\beta = 1.1$. For a detailed description and results of FSA, refer to Khan and Engelbrecht [31].

Table 13 summarizes the results of FSA and FPSO. The table shows results for the best parameter combination for FPSO. For FSA, the best results along with the corresponding cooling rate α are also given in the table. It is observed from Table 13 that the average overall goodness found by the two algorithms are more or less in the same ranges. FPSO had a slight deterioration of 0.6% and 1.74% in the average overall goodness for cases *n15* and *n25* respectively. For *n33*, FPSO had a mild improvement of 0.29% over FSA. A two-sided t-test was also performed to test the hypothesis whether the two averages (i.e. average overall goodness

found by FSA and FPSO) were significantly different from each other. The t-test results were obtained at a 5% significance level. The results showed that both FSA and FPSO produced results of the same quality for *n15*, *n25*, and *n33*. However, for *n40* and *n50*, the deterioration was statistically significant (as validated by the t-test) with 6.15% and 4% respectively.

Table 13 also lists the execution time for FSA and FPSO. It is observed from the table that the execution time for FPSO for cases *n15*, *n25*, and *n33* was quite less than that of FSA. For *n40*, FPSO had a slightly higher execution time than that of FSA, while for *n50*, FPSO had a significantly higher execution time than that of FSA. Thus, the general observation is that as far as computational time is concerned, FPSO had much better performance than FSA for small and medium size test cases, but for large size cases, FSA demonstrated superior performance compared to FPSO.

6.6 Comparison with a Fuzzy Ant Colony Optimization Algorithm

The FPSO algorithm was also compared with a fuzzy ant colony optimization (FACO) algorithm for the DLAN topology design problem [32]. FACO is a multi-objective optimization algorithm based on the ant colony optimization (ACO) meta-heuristic. ACO is another swarm intelligence technique and maintains a population of ants, where each ant is responsible for building a feasible network topology. The ant starts with the root node and incrementally builds a topology. The guiding factors in the process of decision and selection of a particular path are the heuristic value, pheromone deposit, and pheromone evaporation. A complete tour by an ant results in a complete feasible network topology. A detailed description and analysis of FACO can be found in [32].

Table 14 shows a comparison of the best results produced by FPSO and FACO. Columns 2 to 4 show the parameter setup that resulted in the best solutions (best average goodness) for FACO. More specifically, column 2 shows the number of ants, while columns 3 and 4 display the deposit and evaporation rates respectively. Column 5 shows the best average overall goodness produced by FACO. Similarly, columns 6 to 9 give the FPSO parameters that resulted in best average overall goodness, given in column 10. The last column of Table 14 reports the percentage improvement obtained by FACO compared to FPSO. The percentage improvement reflects the percentage difference between the average overall goodness obtained by FACO and that by FPSO. The results in the last column suggest that FACO was able to achieve statistically significant improvement (as validated by a t-test) for two cases (*n25* and *n33*), while for the remaining three cases (*n15*,

Table 14 Comparison of FACO and FPSO for UAO. ants = number of ants dep = pheromone deposit rate, evap = pheromone evaporation rate, par = number of particles, Time = average execution time in seconds, and % imp = percentage improvement achieved by FACO. OG = overall goodness. Statistically significant improvement is in italics.

Case	FACO					FPSO						% imp FACO vs FPSO
	ants	dep	evap	OG	Time	par	V_{max}	c_1, c_2	w	OG	Time	
n15	30	0.8	0.3	0.334 ± 0.002	31.3	20	5	1.49	0.72	0.333 ± 0.002	29.7	0.30
n25	30	0.4	0.1	0.363 ± 0.008	268.5	30	5	0.5	0.72	0.339 ± 0.008	187.0	*6.61*
n33	25	0.8	0.3	0.349 ± 0.006	528.1	20	5	0.5	0.4	0.340 ± 0.008	428.1	*2.58*
n40	30	0.6	0.2	0.352 ± 0.006	1561.9	30	5	0.5	0.72	0.351 ± 0.012	1659.2	0.28
n50	30	0.6	0.2	0.336 ± 0.004	6478.8	30	5	0.5	0.72	0.336 ± 0.003	7074.9	0.0

n40, and *n50*), both FACO and FPSO showed equal performance as there was no statistically significant difference in the results. Therefore, it can be fairly claimed that overall, FACO and FPSO demonstrated more or less equal performance.

Table 14 also lists the execution time for FACO and FPSO. Note that both FPSO and FACO were run for 100 generations. It is observed from the table that the execution time for FPSO for case *n15* was slightly less than that of of FACO. For *n25* and *n33*, FPSO had a far less execution time than that of FACO. For *n40*, FPSO had a slightly higher execution time than that of FACO, while for *n50*, FPSO had considerably higher execution time compared to FACO. Thus, the general observation is that as far as computational time is concerned, FPSO had a much better performance than FACO for small and medium size test cases, but FACO demonstrated better performance compared to FPSO for large size test cases.

7 Conclusions

A fuzzy multi-objective particle swarm optimization algorithm for the DLAN topology design problem was proposed and investigated in this paper. The performance of the algorithm was evaluated with respect to different parameters of the fuzzy PSO algorithm. Results showed that the larger swarm sizes produced better results than medium or small sizes. An investigation of acceleration coefficients revealed that there was no significant difference in the quality of final solutions obtained with respect to the three sets of values of acceleration coefficients used. Results also suggested that the fuzzy PSO was insensitive to the inertia weight, with respect to the three values used. As for velocity clamping, the results suggested that the parameter did not have a significant effect on the quality of the solutions with the three values used. A comparison of the fuzzy PSO with the fuzzy simulated annealing algorithm showed that the fuzzy PSO produced results of statistically equal or slightly inferior quality. Furthermore, comparison with a fuzzy ant colony optimization algorithm suggested that the fuzzy PSO also had results of statistically equal or slightly inferior quality than fuzzy ACO. In near future, we intend to study the effects of PSO parameters in more depth, and to compare fuzzy PSO with other techniques. Furthermore, since fuzzy PSO deals with discrete valued variables, it is also our intention to compare the fuzzy PSO algorithm with other variants of discrete PSO to assess the effectiveness of fuzzy PSO.

References

1. Al-Jaafreh M, Al-Jumaily M (2006) Particle Swarm Optimization based Stroke Volume Influence on Mean Arterial Pressure. In: Proceedings of the IEEE International Conference on Biomedical and Pharmaceutical Engineering, pp 508-512
2. Aggarwal KK, Rai S (1981) Reliability evaluation in computer communication networks. IEEE Transactions on Reliability, 30(1):32-35
3. Angeline P (1998) Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences. V. W. Porto, N. Saravanan, D. Waagen, and A. Eiben (Eds.), Evolutionary Programming VII, Springer, pp 601-610.
4. Atiqullah MM, Rao SS (1993) Reliability optimization of communication networks using simulated annealing. Microelectron Reliability, 33(9):1303-1319
5. Bartz-Beielstein T, Limbourg P, Parsopoulos K, Vrahatis M, Mehnen J, Schmitt K (2003) Particle Swarm Optimizers for Pareto Optimization with Enhanced Archiving Techniques. IEEE Congress on Evolutionary Computation, pp 1780-1787
6. Baumgartner U, Magele C, Renhart W (2004) Pareto Optimality and Particle Swarm Optimization. IEEE Trans Magn 40(2): 1172-1175
7. Chow C, Tsui H (2004) Autonomous Agent Response Learning by a Multi-species Particle Swarm Optimiza-

tion. In: Proceedings of IEEE Congress on Evolutionary Computation, pp 778-785

8. Cho H, Wang B, Roychowdhury S (1998) Automatic Rule Generation for Fuzzy Controllers using Genetic Algorithms: A Study on Representation Scheme and Mutation Rate. In: Proceedings of IEEE World Congress on Computational Intelligence, pp 1290-1295
9. Coello-Coello CA, Lechuga M (2002) MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, pp 1051-1056
10. Dengiz B, Altiparmak F, Smith A (1997) Local Search Genetic Algorithm for Optimal Design of Reliable Network. IEEE Trans Evol Comput, 1:179-188
11. Eberhart R, Kennedy J (1995) A New Optimizer using Particle Swarm Theory. In: Proceedings of the 6th International Symposium on Micro Machine and Human Science, pp 39-43
12. Eberhart R, Kennedy J (1995) Particle Swarm Optimization. In: Proceedings of the IEEE International Joint Conference on Neural Networks, pp 1942-1948
13. Eberhart R, Kennedy J (1999) The Particle Swarm: Social Adaptation in Information Processing Systems. D. Corne, M. Dorigo, and F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, pages 379-387
14. Eberhart R, Simpson P, Dobbins R (1996) Computational Intelligence PC Tools. Academic Press
15. Elbaum R, Sidi M (1996) Topological Design of Local-Area Networks Using Genetic Algorithm. IEEE/ACM Trans Netw, 4:766-778
16. Engelbrecht AP (2005) Fundamentals of Computational Swarm Intelligence. John Wiley Sons
17. Ersoy C, Panwar S (1993) Topological Design of Interconnected LAN/MAN Networks. IEEE J Sel Area Commun, 11:1172-1182
18. Esau LR, Williams KC (1966) On teleprocessing system design. A method for approximating the optimal network. IBM Syst J, 5:142-147
19. Fieldsend F, Singh S (2002) A Multiobjective Algorithm Based Upon Particle Swarm Optimisation, An Efficient Data Structure and Turbulence, In: Proceedings of U.K. Workshop on Computational Intelligence, pp 37-44
20. Gen M, Ida K, Kim J (1998) A Spanning Tree-Based Genetic Algorithm for Bicriteria Topological Network Design. In: Proceedgins of IEEE International Conference on Evolutionary Computation, pp 164-173
21. Habib S (2005) Redesigning Network Topology with Technology Considerations. In: Proceedings of the 9th IFIP/IEEE International Symposium on Integrated Network Management, pp 207-219
22. Hamacher H (1978) Ueber logische Verknupfungen Unschalfer Aussagen und deren Zugehoerige Bewertungs-funktione. Prog Cybern Syst Res, 3:276-288.
23. Haupt R (2000) Optimum Population Size and Mutation Rate for a Simple Real Genetic Algorithm that Optimizes Array Factors. In: Proceedings of IEEE Antennas and Propagation Society International Symposium, pp 1034 - 1037
24. Ho SL, Shiyou Y, Guangzheng N, Lo E, Wong H (2005) A Particle Swarm Optimization Based Method for Multiobjective Design Optimizations. IEEE Trans Magn 41(5): 1756-1759
25. Hu X, Eberhart R, Shi Y (2003) Particle Swarm with Extended Memory for Multiobjective Optimization.

In: Proceedings of IEEE Swarm Intelligence Symposium, pp 193-197

26. Hu X, Eberhart R (2002) Multiobjective Optimization using Dynamic Neighborhood Particle Swarm Optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, pp 1677-1681
27. Ince T, Kiranyaz S, Gabbouj M (2009) A Generic and Robust System for Automated Patient-Specific Classification of ECG Signals. IEEE Trans Biomed Eng 56(5):1415-1426
28. Keiser GE (1989) Local Area Networks. McGraw-Hill Book Company
29. Kershenbaum A (1993) Telecommunications Network Design Algorithms. McGraw-Hill Publishers, USA
30. Khan SA, Engelbrecht AP (2007) A new fuzzy operator and its application to topology design of distributed local area networks, Inf Sci, 177:2692-2711
31. Khan SA, Engelbrecht AP (2009) Fuzzy Hybrid Simulated Annealing Algorithms for Topology Design of Switched Local Area Networks. Soft Comput 3(1):45-61
32. Khan SA, Engelbrecht AP (2008) A Fuzzy Ant Colony Optimization Algorithm for Topology Design of Distributed Local Area Networks. In: Proceedings of IEEE Swarm Intelligence Symposiumn, pp 1-7
33. Kirkpatrick S, Gelatt C, Vecchi M (1983) Optimization by Simulated Annealing. Sci, pp 498-516
34. Kruskal JB (1956) On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. Am Math Soc 7(1):48-50
35. Kumar A, Pathak M, Gupta Y (1995) Genetic Algorithm-Based Reliability Optimization for Computer Network Expansion. IEEE Trans Reliab, 24:63-72
36. Li H, Yen V (1997) Fuzzy sets and fuzzy decision-making. Kluwer Publishers
37. Lim M, Rahardja S, Gwee B (1996) A GA Paradigm for Learning Fuzzy Rules. Fuzzy Sets Syst 82:177-186
38. Liska J, Melsheimer S (1994) Complete Design of Fuzzy Login System using Genetic Algorithms. In: Proceedings of 3rd IEEE International Conference on Fuzzy Systems, pp 1377-1382
39. Miettinen K (2001) Some Methods for Nonlinear Multi-objective Optimization. In: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, LNCS, pp 1 - 20
40. Moore J, Chapman R (1999) Application of Particle Swarm to Multiobjective Optimization. Technical Report, Department of Computer Science and Software Engineering, Auburn University
41. Mostaghim S, Teich J (2003) Strategies for Finding Good Local Guides in Multi-objective Particle Swarm Optimization. In: Proceedings of IEEE Swarm Intelligence Symposium. pp 26-33
42. Ochoa A, Hernandez A, Gonzalez S, Jons S, Padilla A (2008) Hybrid System to Determine the Ranking of a Returning Participant in Eurovision. In: Proceedings of the IEEE 8th International Conference on Hybrid Intelligent Systems, pp 489-494
43. Parsopoulos K, Tasoulis D, Vrahatis M (2004) Multiobjective Optimization using Parallel Vector Evaluated Particle Swarm Optimization. In: Proceedings of IASTED International Conference on Artificial Intelligence and Applications, pp 823-828
44. Parsopoulos K, Vrahatis M (2002) Recent Approaches to Global Optimization Problems through Particle Swarm Optimization. Nat Comput 1:235-306

45. Parsopoulos K, Vrahatis M (2002) Particle Swarm Optimization Method in Multiobjective Problems. In: Proceedings of ACM Symposium on Applied Computing, pp 603-607

46. Prim RC (1957) Shortest Connection Networks and Some Generalizations. Bell Syst Tech J 36:1389-1401

47. Ray T, Liew KM (2002) A Swarm Metaphor for Multiobjective Design Optimization. Eng Optim 34(2):141-153

48. Reeves W (1983) Particle Systems - A Technique for Modelling a Class of Fuzzy Objects. ACM Trans Graph 2(2):91-108

49. Reyes-Sierra M, Coello-Coello CA (2006) Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. Int J Comput Intell Res 2(3):287-308

50. Shi Y, Eberhart R (1998) Parameter Selection in Particle Swarm Optimization. V. W. Porto, N. Saravanan, D. Waagen, and A. Eiben (Eds.), Evolutionary Programming VII, Springer, pp 611-616.

51. Sportack MA (1999). IP Routing Fundamentals. Cisco Press

52. Van den Bergh F (2001) An Analysis of Particle Swarm Optimizers. PhD Thesis, University of Pretoria

53. Van den Bergh F, Engelbrecht AP (2001) Training Product Unit Networks using Cooperative Particle Swarm Optimizers. In: Proceedings of IEEE International Joint Conference on Neural Networks, pp 126-132

54. Van den Bergh F, Engelbrecht AP (2002) A New Locally Convergent Particle Swarm Optimizer. In: Proceedings of IEEE Conference on Systems, Man, and Cybernetics, pp 96-101

55. Wang L, Singh C (2006) Stochastic Combined Heat and Power Dispatch based on Multi-objective Particle Swarm Optimization. In: Proceedings of the IEEE Power Engineering Society General Meeting, pp 1-8

56. Xu C, Zhang Q, Wang B, Zhang R (2008) Improved Particle Swarm Optimization Algorithm for 2D Protein Folding Prediction. In: Proceedings of the IEEE 2nd International Conference on Bioinformatics and Biomedical Engineering, pp 816-819

57. Yager R (1988) On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking. IEEE Trans Syst Man Cybern, 18(1):183-190

58. Yoshida H, Kawata K, Fukuyama Y, Takayama S, Nakanishi Y (2000) A Particle Swarm Optimization for Reactive Power and Voltage Control considering Voltage Security Assessment. IEEE Trans Power Syst 15(4):1232-1239

59. Youssef H, Sait S, Issa O (1997) Computer-Aided Design of Structured Backbones. In: Proceedings of 15th National Computer Conference and Exhibition, pp 1-18

60. Youssef H, Sait S, Khan SA (2002) Topology design of switched enterprise networks using a fuzzy simulated evolution algorithm. Eng Appl Artif Intell, 15:327-340

61. Youssef H, Sait S, Khan SA (2000) Fuzzy Simulated Evolution Algorithm for Topology Design of Campus Networks. In: Proceedings of IEEE Congress on Evolutionary Computation, pp 180-187

62. Youssef H, Sait S, Khan SA (2004) A Fuzzy Evolutionary Algorithm for Topology Design of Campus Networks. Arab J Sci Eng 29(2b):195-212

63. Zadeh LA (1963) Optimality and Non-Scalar-Valued Performance Criteria. IEEE Trans Autom Contr, 8:59-60

64. Zadeh LA (1965) Fuzzy Sets. Inf Contr, 8:338-353

65. Zhang X, Li T (2007) Improved Particle Swarm Optimization Algorithm for 2D Protein Folding Prediction. In: Proceedings of the IEEE 1st International Conference on Bioinformatics and Biomedical Engineering, pp 53-56