Fuzzy Particle Swarm Optimization Algorithms for the Open Shortest Path First Weight Setting Problem

Mohammad A. Mohiuddin · Salman A. Khan · Andries P. Engelbrecht

Abstract The open shortest path first (OSPF) routing protocol is a well-known approach for routing packets from a source node to a destination node. The protocol assigns weights (or costs) to the links of a network. These weights are used to determine the shortest paths between all sources to all destination nodes. Assignment of these weights to the links is classified as an NP-hard problem. The aim behind the solution to the OSPF weight setting problem is to obtain optimized routing paths to enhance the utilization of the network.

Mohammad A. Mohiuddin

Department of Computer Science, University of Pretoria, Pretoria 0002, South Africa E-mail: waseem_aijaz@yahoo.com

Salman A. Khan

Department Computer Engineering Department, College of IT, University of Bahrain, Sakhir, Bahrain

Department of Computer Science, University of Pretoria, Pretoria 0002, South Africa E-mail: sakhan@uob.edu.bh, skhan@cs.up.ac.za

Andries P. Engelbrecht

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

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

This paper formulates the above problem as a multi-objective optimization problem. The optimization metrics are maximum utilization, number of congested links, and number of unused links. These metrics are conflicting in nature, which motivates the use of fuzzy logic to be employed as a tool to aggregate these metrics into a scalar cost function. This scalar cost function is then optimized using a fuzzy particle swarm optimization (FPSO) algorithm developed in this paper. A modified variant of the proposed PSO, namely, fuzzy evolutionary PSO (FEPSO), is also developed. FEPSO incorporates the characteristics of the simulated evolution heuristic into FPSO. Experimentation is done using 12 test cases reported in literature. These test cases consist of 50 and 100 nodes, with the number of arcs ranging from 148 to 503. Empirical results have been obtained and analyzed for different values of FPSO parameters. Results also suggest that FEPSO outperformed FPSO in terms of quality of solution by achieving improvements between 7% and 31%. Furthermore, comparison of FEPSO with various other algorithms such as Pareto-dominance PSO, weighted aggregation PSO, NSGA-II, simulated evolution, and simulated annealing algorithms revealed that FEPSO performed better than all of them by achieving best results for two or all three objectives.

Keywords Open Shortest Path First Routing Algorithm, Particle Swarm Optimization, Swarm Intelligence, Multi-objective Optimization, Fuzzy Logic

1 Introduction

The use of web based applications has resulted in rapid increase of internet traffic [1]. Efficient utilization of network resources, such as network bandwidth, is essential to deal with this high volume of traffic. The main objective of network traffic engineering is efficient mapping of traffic on the available network resources to prevent traffic imbalance, if it exists [2].

Routers serve as the main interconnection points of the internet and forward data packets between source and destination nodes via multiple paths. These paths exist between a given source and destination pair. The internet is huge and very complex and is divided into autonomous systems (AS) for managing its complexity. An AS represents a collection of networks under the control of one single entity or organization with a specific routing policy. These policies are determined by a class of routing protocols, namely, interior gateway protocols (IGPs) [3]. Routing across ASs is performed by another class of protocols, namely, exterior gateway protocols (EGPs) [3].

Open shortest path first (OSPF) [3] is an IGP and has received notable attention by researchers for efficient traffic engineering since OSPF is considered the best backbone routing protocol used in the internet [4, 5]. The protocol has shown remarkable performance through significant reduction in maximum utilization over pure shortest path routing [6]. OSPF is based on Dijkstra's algorithm [7], which determines a shortest path between a source and destination pair. Each link in the network is given a measurable entity called a link weight or OSPF weight. The cost of a path between a given source and destination pair is found by the summation of OSPF weights on the links in that path. The path with minimal cost is labelled as the shortest path.

This paper considers the open shortest path first weight setting (OSPFWS) problem, classified as an NP-hard problem [8]. The OSPFWS problem requires a set of weights to be determined, so as to efficiently utilize network resources. The objectives of this problem are to minimize maximum utilization, minimize the number of congested links, and to minimize the number of unused links. These objectives conflict with each other, i.e. if one objective is improved, at least one of the other objectives may deteriorate. To address this NP-hard problem with conflicting objectives, this paper proposes to apply a fuzzy particle swarm optimization (FPSO) algorithm. The paper also proposes a hybrid PSO, namely, fuzzy evo-

lutionary PSO, where characteristics of simulated evolution algorithm [9] are combined with the fuzzy PSO. The performance of these two variants is empirically assessed and compared.

The rest of the paper is organized as follows: Section 2 provides the necessary background and related work on the OSPFWS problem. Section 3 provides the formal definition of the OSPFWS problem. A brief discussion on fuzzy logic and the Unified And-OR operator is given in Section 4. Section 5 describes the formulation of a fuzzy logic based objective function for the OSPFWS problem. Section 6 presents the proposed fuzzy PSO algorithm, and a variant of the fuzzy PSO, the fuzzy evolutionary PSO, is proposed and discussed in Section 7. The experimental methodology is described in Section 8. Results are provided and discussed in Section 9. A comparative analysis of the fuzzy evolutionary PSO with other algorithms, namely, Pareto-dominance PSO, weighted aggregation PSO, NSGA-II, simulated evolution, and simulated annealing is provided in Section 10. The paper is concluded in Section 11. Finally, the symbols and terminology used in this paper are given in Appendix A. Some additional results related to the analysis of swarm size (discussed in Section 9) are provided in Appendix B.

2 Literature review

Notable research in optimizing OSPF weights has been reported in the literature [2,4,6, 10–28]. The pioneering work on the OPSF weight setting problem was done by Fortz and Thorup [8,10,29] who used maximum utilization as the optimization objective. The term "maximum utilization" refers to the maximum of all utilization values over all the links in the network. A cost function based on utilization ranges was first formulated by Fortz and Thorup [2], who applied tabu search [30] to minimize "maximum utilization".

The cost function of Fortz and Thorup was formally defined as

$$minimize \quad \Phi = \sum_{a \in A} \Phi_a(l_a) \tag{1}$$

subject to the constraints:

$$l_a = \sum_{(s,t) \in N \times N} f_a^{(s,t)} \; a \in A, \tag{2}$$

$$f_a^{(s,t)} \geq 0 \tag{3}$$

where Φ is the cost function, Φ_a is the cost associated with arc a, l_a is the total traffic load on arc a, $f_a^{(s,t)}$ represents traffic flow from node s to t over arc a, N defines the set of nodes, and A represents the set of arcs. Equation (2) indicates that the total load (traffic) on arc a is equal to the sum of the traffic load on arc a and the traffic load on all incoming arcs to arc a. The constraint in Equation (3) implies that the traffic flow from node s to t over arc a can be greater than or equal to zero.

In Equation (1), Φ_a represents piecewise linear functions, with $\Phi_a(0) = 0$ and a derivative, $\Phi_a^{'}(l_a)$ given by

$$\Phi_a^{'}(l) = \begin{cases} 1 & \text{for } 0 \leq l/c_a < 1/3, \\ 3 & \text{for } 1/3 \leq l/c_a < 2/3, \\ 10 & \text{for } 2/3 \leq l/c_a < 9/10, \\ 70 & \text{for } 9/10 \leq l/c_a < 1, \\ 500 & \text{for } 1 \leq l/c_a < 11/10, \\ 5000 & \text{for } 11/10 \leq l/c_a < infinity \end{cases} \tag{4}$$

The above function indicates that the utilization (which represents the load to capacity ratio) of a link is acceptable within 100% of the link's capacity. According to the function in Equation (4), links with utilization levels less than or equal to 1 (or 100%) have a low cost, proportional to the level of utilization. These values are 1, 3, 10, or 70. Furthermore, links exceeding 100% utilization are assigned high costs of 500 and 5000. For example, if utilization is less than one third of a link's capacity, then a cost of 1 is assigned. For utilization between 1/3 and 2/3 of a link's capacity, a cost of 3 is assigned, and so on. On the other hand, if the utilization of a link is beyond 1 (which indicates that the number of incoming packets to a link exceed the maximum capacity of the link) then such an over-utilization is not desirable, since it will result in packet loss. Therefore, the cost assigned to the links beyond 100% utilization is much higher (i.e. 500 for utilization of equal to or more than 100% but less than 110%, and 5000 for more than 110%). Note that as per Equation (4), a link with utilization greater than 100% and less than 110% is still preferable compared to a link with utilization greater than 110%. Fortz and Thorup employed a dynamic shortest path algorithm [31–33] to obtain multiple equidistant shortest paths between a source-destination pair. By this mechanism, traffic load was distributed equally across the links.

Subsequent to the work of Fortz and Thorup, many other researchers attempted to solve the OSPF weight setting problem with different algorithms and different objective functions. Ramakrishnan and Rodrigues [12] proposed a local search procedure using the same cost function as that of Fortz and Thorup. The main difference between the two approaches was that for a heavily used link, Rodrigues and Ramakrishnan's technique increases the link metric (i.e. the OSPF weight assigned to a link). Ericsson *et al.* [13] developed a genetic algorithm [34] to solve the OSPFWS problem also using the cost function by Fortz and Thorup. Kandula *et al.* [6] compared the performance of three OSPF weight optimizers while considering maximum link utilization as the optimization objective. Bhagat [4]

also assumed link utilization as weights and used a genetic algorithm for OSPF weight setting while using the cost model of Fortz and Thorup with a minor modification. Abo Ghazala *et al.* [35] performed a survey of various algorithms applied to the OSPFWS problem, and also proposed a technique based on iterative local search, while considering link utilization as the optimization objective. The underlying cost function was the same as proposed by Fortz and Thorup. In a subsequent research article, Aboghazala *et al.* [15] assumed maximization of unused bandwidth as the optimization objective and employed simulated annealing and hybrid genetic algorithms for weight optimization. Parmar *et al.* [16] formulated the OSPF weight setting problem as mixed-integer linear programming problem and developed a branch-and-cut algorithm while assuming minimization of network congestion as the optimization objective. Pioro *et al.* [22] considered the maximum load on any link in the network as the measure of congestion and proposed two heuristic approaches for weight setting. Srivastava *et al.* [17] also considered minimization of maximum load on any link and proposed heuristic algorithms based on Lagrangian relaxation to determine feasible solutions for the weight setting problem. Buriol *et al.* [18] extended the genetic algorithm proposed in [13] to a memetic algorithm by adding a local search procedure while using the same cost function as that of Fortz and Thorup. Bley [19,20] proposed unsplittable shortest path routing (UPSR) and claimed that the proposed approach can be applied to other routing schemes such as OSPF, while considering minimization of maximum congestion over all arcs. Zagozdzon *et al.* [14] proposed a two-phase algorithm for resolving the OSPF weight setting problem while considering the residual capacity as the optimization objective. This residual capacity resulted from setting the link weights proportional to the inverse of their capacity. Reis *et al.* [36] proposed a memetic algorithm for weight setting in OSPF and DEFT algorithms while considering minimization of total link utilization. Lin and Gen [21] proposed a priority-based genetic algorithm for shortest path routing in OSPF. Their results

indicated that the proposed GA could be used for weight setting in OSPF and other routing algorithms. Retvari *et al.* [23,24] studied the OSPF weight setting problem considering maximization of network throughput and proposed some algorithms that could efficiently optimize the link weights. Nucci *et al.* [25] proposed a Tabu-search heuristic for choosing link weights that takes into account both service level agreement (SLA) requirements and link failures with the objective of optimization link utilizations. Shirmali *et al.* [26] devised an approach based on *Nash bargaining and decomposition*. It was claimed that the proposed approach could be easily modified to yield a mechanism for setting link weights for ISPs using OSPF in a way similar to that of Fortz and Thorup. Riedl [27] presented an algorithm based on simulated annealing to optimize link metrics in OSPF networks. The algorithm took into account the original routing configuration and allowed tradeoff considerations between routing optimality and adaptation impact. Lee *et al.* [28] modelled the optimal link weight assignment problem as an integer linear programming problem while considering minimization of sum of energy consumption of all links.

It is noteworthy of mentioning that, generally, the aforementioned approaches considered a single objective in the optimization process. For example, the cost function proposed by Fortz and Thorup (Equation (1)) on which many subsequent attempts were based [12,13, 37,4,35,18,26] considered minimization of maximum link utilization. Other researchers [6, 22,17] also assumed minimization of maximum link utilization. Other objectives considered in the optimization process were maximization of unused link bandwidth [15], minimization of network congestion [16,19,20], residual capacity of link [14], minimization of total link utilization [36], maximization of network throughput [23,24], and minimization of sum of energy consumption of all links [28]. Exceptions from these single-objectives optimization approaches were Nucci *et al.* [25] where link failure and link no-failure states were used as the optimization objectives, and Sqalli *et al.* [11,38] who used minimization of maximum

link utilization as well as minimization of congested links as the optimization objectives while using a simulated annealing (SA) algorithm [39].

A cost function developed by Sqalli *et al.* [11,38] evolved from the earlier work by Fortz and Thorup. The reason for using the cost function of Fortz and Thorup was that the function was employed in many studies as mentioned above. One novel aspect of the work of Sqalli *et al.* was the addition of another optimization objective (i.e. minimization of congested link) on top of minimization of maximum link utilization. This resulted in better distribution of traffic in the network since this is one fundamental requirement of network traffic engineering. The function employed by Sqalli *et al.* is defined as

$$\Phi = MU + \frac{\sum_{a \in SetCA} (l_a - c_a)}{E} \tag{5}$$

where MU is the maximum utilization of the network. $SetCA$ defines the set of congested links, E represents the total number of links in the network, c_a refers to the capacity of link a, and l_a is the total traffic on link a.

The second term of Equation (5) after the plus sign defines the extra load on the network. This extra load is found by taking all the congested links, divided by the total number of links present in the network to normalize the entire function. For an uncongested network, the term after the plus sign results in a zero. Thus, Equation (5) results in minimization of maximum utilization provided that there is no congestion in the network. If congestion exists, then the function results in the minimization of maximum utilization as well as the minimization of the number of congested links. Sqalli *et al.* concluded that the cost function in Equation (5) results in more efficient minimization of the number of congested links compared to the cost function of Fortz and Thorup [2]. Furthermore, Sqalli *et al.* discovered that the results for maximum utilization with their method were comparable to those obtained

by the approach of Fortz and Thorup. Using the cost function of Equation (5), Sqalli *et al.* applied the simulated evolution (SimE) algorithm [40] to the OSPFWS problem and compared the results with the results of SA [38]. Tabu search using the cost function of Sqalli *et al.* [11] has also been applied to the OSPFWS problem [41].

A limitation of the cost function of Fortz and Thorup is that it minimizes "maximum utilization" only. This may lead to the existence of links which are either congested or unused. The cost function proposed by Sqalli *et al.* (Equation (5)) was aimed at simultaneous optimization of maximum utilization and the number of congested links, without any consideration of unused links. It is, therefore, not guaranteed that optimizing maximum utilization and number of congested links would implicitly optimize the number of unused links as well. This observation points to the fact that to have a more stable traffic flow, traffic from congested links should be shifted to unused links. Therefore, in order to overcome this issue, Mohiuddin *et al.* [42] proposed a fuzzy logic based cost function that addresses the simultaneous optimization of maximum utilization, number of congested links, and number of unused links through fuzzy logic based aggregation. Mohiuddin *et al.* used their fuzzy cost function with three iterative heuristics, namely, simulated evolution, simulated annealing, and NGSA-II, and performed a mutual comparison of the three algorithms.

3 Open Shortest Part First Weight Setting Problem Definition

This section provides the details of the OSPFWS problem. More specifically, the section provides a formal definition of the OSPFWS problem, followed by a discussion of the calculation of traffic load on links.

3.1 Open Shortest Path First Weight Setting Problem

The OSPFWS problem is formulated as follows. Given a network topology and predicted traffic demands, find a set of OSPF weights that optimizes network performance. More precisely, given a directed network $G = (N, A)$, a demand matrix D, and capacity C_a for each arc $a \in A$, determine a positive integer weight $q_a \in [1, q_{max}]$ for each arc $a \in A$ such that the objective function or cost function Φ is minimized. The maximum value of this weight, q_{max}, is a user-defined upper limit. Fortz and Thorup [29] discovered that a small set of weight values significantly reduces the overhead of the algorithm. By experimentation, they set w_{max} to 20. The chosen weights on arcs determine the shortest paths, which in turn completely determine the routing of traffic flow, the loads on the arcs, and the value of the cost function. The quality of OSPF routing is highly dependent on the selection of weights. Figure 1 shows a topology with weights assigned to each arc. These weights are in the range $[1, 20]$. A solution for this topology can be $(18, 1, 7, 15, 3, 17, 14, 19, 13, 18, 4, 16, 16)$. The weights are arranged through a breadth-first traversal of the graph. For example, for node A, the weights on the outgoing links are 18 and 1. For node B, the weights on outgoing links are 7 and 15, and so on.

For the purposes of this paper, three objectives are considered. These objectives are maximum utilization, number of congested links, and number of unused links, all of which need to be minimized simultaneously. Minimizing maximum utilization will lead to better distribution of network traffic across all the links such that congestion can be avoided and the network can be utilized well as per its capacity [8]. Network administrators desire less congested links. However, if a network is highly congested, then the preference is to reduce the congestion by at least minimizing the total number of congested links. For example, assume a network with 50 congested links and 20 unused links. It would be preferred to

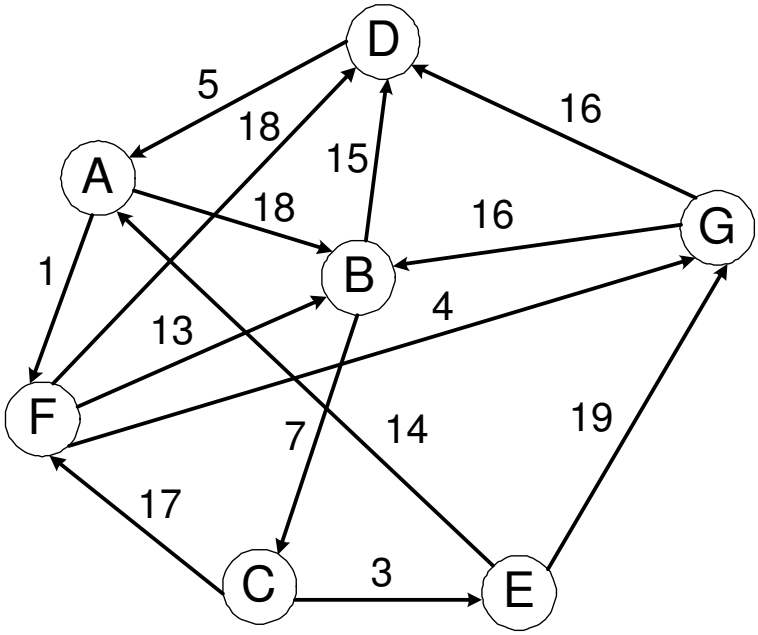

Fig. 1 Representation of a topology with assigned weights.

accommodate the traffic of the 50 congested links additionally on the 20 unused links. This indicates that minimizing the number of unused links also affects the performance of the network. This positive effect on the performance is due to traffic distribution across the links of the networks which depends on the routing paths established [42]. Therefore, a new solution might create new routing paths such that traffic on congested links may be distributed on unused links.

3.2 Traffic Load Calculation

This section provides details of the steps to calculate arc (or link) loads. Given a weight setting $\{w_a\}_{a \in A}$, the arc loads l_a are calculated in five steps. For all demand pairs $d_{st} \in D$, consider one destination t at a time and compute partial arc loads $l_a^t \; \forall \, t \in \bar{N} \subseteq N$, where $\bar{N}$ is the set of destination nodes. The steps are as follows:

1. Compute the shortest distances d_u^t from each node $u \in N$ to t, using Dijkstra's shortest path algorithm [7]. Dijkstra's algorithm usually computes the distances away from source s, but since it is required to compute the distance to the sink node t, the algorithm is applied on the graph obtained by reversing all arcs in G.
2. Compute the set A^t of arcs on shortest paths to t as,

$$A^t = \{(u,v) \in A : d_u^t - d_v^t = w_{(u,v)}\}$$

3. For each node u, let δ_u^t denote its outdegree in $G^t = (N, A^t)$, i.e.,

$$\delta_u^t = \mid \{v \in N : (u,v) \in A_t\} \mid$$

   If $\delta_u^t > 1$, then traffic flow is split at node u to balance the load.
4. The partial loads l_a^t are computed as follows:
   (a) Nodes $v \in N$ are visited in order of decreasing distance d_v^t to t.
   (b) When visiting a node v, for all $(v,w) \in A^t$, set

$$l_{(v,w)}^t = 1/[\delta_v^t(d_{vt} + \textstyle\sum_{(u,v)\in A^t} l_{(u,v)}^t)]$$

5. The arc load l_a is now summed from the partial loads as:

$$l_a = \textstyle\sum_{t\in\bar{N}} l_a^t$$

4 Fuzzy Logic and Aggregation Operators

In general terms, a crisp set X is defined as a collection of objects $x \in X$, where each object can either belong to the set or not. However, in many practical situations, certain objects do not fulfil this "crisp" membership requirement. In such situations, a need arises for another set theory which could deal with uncertain data. One possible approach is fuzzy set theory (FST), which aims to represent vague information.

The basis of the theory of fuzzy sets [43,44] is multi-valued logic wherein a statement can be partly false and partly true at the same time. Formally, a fuzzy set is characterized by a membership function, μ, in the range [0,1]. The membership function provides a measure of the degree of presence for every element in the set [45]. A value of $\mu = 1$ indicates that the statement is true, while $\mu = 0$ indicates that the statement is false.

Similar to crisp sets, set operations such as union, intersection, and complement are also defined on fuzzy sets. A number of operators exist for fuzzy union and fuzzy intersection. Fuzzy intersection operators are referred to as t-norm operators while fuzzy union operators are known as s-norm operators. Generally, the t-norm is implemented using "min" and the s-norm using "max". However, in the formulation of multi-criteria decision functions, the simple AND (pure "min" function) and simple OR (pure "max" function) does not work well, due to the fact that the simple AND or OR operations consider the effect of only one objective while neglecting the effects of other objectives. This deficiency of the simple AND and simple OR operators resulted in the development of a number of "soft-AND" and "soft-OR" operators, such as the Werners operator [46], Einstein's operator [46], Hamacher's operator [47], Frank's operator [48], Weber's operator [49], Dubois and Prade's operator [50], and the Unified And-Or operator [51], among others. These operators allow easy adjustment of the degree of "anding" and "oring" embedded in the aggregation.

Khan and Engelbrecht showed that the Unified And-Or (UAO) operator [51] satisfies the monotonicity, symmetry, and idempotency conditions. One important characteristic of the UAO operator is that a single equation is used to adjust the degree of "anding" and "oring". Yet, the operator is capable of behaving either as the soft-AND or the soft-OR operator. This is in contrast to other aggregation operators listed above, which use separate equations for AND and OR functions. The behavior of ANDing and ORing of UAO is controlled by

a variable, $\nu \geq 0$, whose value decides whether the function behaves as AND or OR. The operator is defined as:

$$f(a,b) = \frac{ab + \nu \max\{a,b\}}{\nu + \max\{a,b\}} = \begin{cases} \mathrm{I}_\star = \mu_{A \cup B}(x) \text{ if } \nu > 1 \\ \mathrm{I}^* = \mu_{A \cap B}(x) \text{ if } 0 \leq \nu \leq 1 \end{cases} \tag{6}$$

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 more details of the UAO operator, the interested reader is referred to Khan and Engelbrecht [51].

5 Fuzzy Logic Approach for the Open Shortest Path First Weight Setting Problem

Although the approach has been previously proposed and explained in Mohiuddin *et al.* [42], it is again summarized below for the sake of completeness. Details can be found in Mohiuddin *et al.* [42].

The solution to the OSPFWS problem is to assign a set of weights to network links. The best solution is one which optimizes the network resources efficiently. The design objectives of the OSPFWS problem include maximum utilization (MU), number of congested links (NOC) and number of unused links (NUL). These objectives individually on their own do not provide adequate information for deciding the quality of a solution. The conflicting nature of these objectives further amplifies the complexity of the problem. With this complexity, a mechanism is required to find a solution that provides the best tradeoff covering all the objectives. Fuzzy logic is one approach that can conveniently and efficiently handle the tradeoff issues between multiple objectives.

The rest of this section details the employment of fuzzy logic for combining the three conflicting objectives into a single overall objective. This overall objective assesses the quality of a solution in terms of membership of a given set of weights. A set of weights providing efficient utilization of network resources consists of low MU, low NOC and low NUL.

To formulate the overall objective function, the values of individual objectives need to be determined first, through membership functions. This needs the formulation of membership functions for each individual objective. This process is described below.

To define the membership function of maximum utilization, two extreme values, the minimum and maximum, are determined first. These values could be found mathematically or from prior knowledge. Figure 2 shows the membership function of the objective to be optimized (maximum utilization in this case). Point 'A' refers to minimum MU (MinMU) and point 'B' refers to maximum MU (MaxMU). The membership value for MU, μ_{MU}, is determined as follows:

$$\mu_{MU}(x) = \begin{cases} 1 & \text{if } MU \leq MinMU \\ \frac{MaxMU - MU}{MaxMU - MinMU} & \text{if } MinMU < MU \leq MaxMU \\ 0 & \text{if } MU > MaxMU \end{cases} \tag{7}$$

The membership function for NOC, μ_{NOC}, is defined in a similar way. In Figure 2, point 'A' then refers to minimum NOC (MinNOC) and 'B' refers to maximum NOC (MaxNOC). The membership function of NOC is defined as follows:

$$\mu_{NOC}(x) = \begin{cases} 1 & \text{if } NOC \leq MinNOC \\ \frac{MaxNOC - NOC}{MaxNOC - MinNOC} & \text{if } MinNOC < NOC \leq MaxNOC \\ 0 & \text{if } NOC > MaxNOC \end{cases} \tag{8}$$

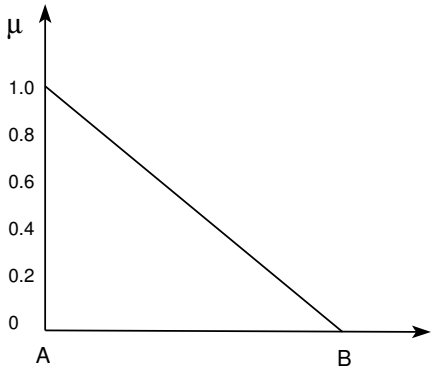

Fig. 2 Membership function of the objective to be optimized

Finally, the membership function for NUL, μ_{NUL}, is defined as

$$\mu_{NUL}(x) = \begin{cases} 1 & \text{if } NUL \leq MinNUL \\ \frac{MaxNUL - NUL}{MaxNUL - MinNUL} & \text{if } MinNUL < NUL \leq MaxNUL \\ 0 & \text{if } NUL > MaxNUL \end{cases} \quad (9)$$

where minimum (MinNUL) and maximum (MaxNUL) values correspond to 'A' and 'B', respectively in Figure 2.

A good solution to the OSPFWS problem is one that is characterized by a low MU, low NOC, and low NUL. In fuzzy logic, this can be stated by the following fuzzy rule:

Rule 1: IF a solution X has low MU AND low NOC AND low NUL THEN it is a good solution.

The words 'low MU', 'low NOC' and 'low NUL' are linguistic values, each defining a fuzzy subset of solutions. Using the UAO operator [51], the above fuzzy rule reduces to the following equation.

$$\mu(x) = \frac{\mu_1(x)\mu_2(x)\mu_3(x) + \nu \times max\{\mu_1(x), \mu_2(x), \mu_3(x)\}}{\nu + max\{\mu_1(x), \mu_2(x), \mu_3(x)\}} \quad (10)$$

where $\mu(x)$ is the membership value for solution x in the fuzzy set "good OSPF Weight set" and ν is a constant in the range [0,1]. Moreover, μ_i for $i = \{1, 2, 3\}$ represents the membership values of solution x in the fuzzy sets low MU, low NOC, and low NUL respectively. The solution which results in the maximum value for Equation (10) is reported as the best solution.

As an example, consider an arbitrary solution S_1, having $\mu_{MU} = 0.19$, $\mu_{NOC} = 0.2$, and $\mu_{NUL} = 0.17$. Also assume that $\nu = 0.5$. Then, Equation (10) results in a value of 0.152. Similarly, consider $\mu_{MU} = 0.22$, $\mu_{NOC} = 0.23$, and $\mu_{NUL} = 0.0.09$ associated with another arbitrary solution S_2. Again assume that $\nu = 0.5$. Then, Equation (10) evaluates to 0.164. Thus, solution S_2 is better than solution S_1 in terms of quality. Equation (10) is employed as a fuzzy cost function for solving the OSPFWS problem using the fuzzy PSO and the fuzzy PSO with simulated evolution algorithms. In this paper, the fuzzy cost function is denoted as FuzzyCF.

6 Fuzzy Particle Swarm Optimization for the Open Shortest Path First Weight Setting Problem

The fuzzy PSO (FPSO) algorithm navigates the search space by maintaining a swarm of candidate solutions, with each candidate solution referred to as a particle. Each particle explores new positions in the search space through its own history, and from the experience of other particles. With respect to the OSPFWS problem, each particle reaches a new candidate solution by changing a few weights on the links of the network. As with the basic PSO [52], the guidance in changing these weights is provided by the particle's current position, its own best position so far, and the global best position obtained so far by the entire algorithm. Each step of the proposed FPSO algorithm is discussed in the following subsections in detail.

6.1 Particle Position and Velocity Representation

The standard PSO uses floating-point vectors to represent positions and velocities. For the OSPFWS problem, this study uses a set representation for particles. Therefore, for an arbitrary network with nodes from a to q, each particle position is defined as a set,

$$\mathbf{X}_i(t) = \{w_{ab}, w_{ac}, ..., w_{aq}, w_{bc}..., w_{pq}\}$$

where w_{ab} is the weight assigned to the link between any two nodes a and b in the network. A constant, W, is also defined as the number of weights in the solution, i.e. $|\mathbf{X}_i(t)| = W$. The velocity of particle i is represented as

$$\mathbf{V}_i(t) = w_{ab} \Leftrightarrow w'_{ab}$$

which represents a sequence of replacements operators where the weight of link (a, b) is replaced with a new value, w'_{ab}, and $|\mathbf{V}_i(t)|$ gives the total number of changes to particle i.

Example 1: Consider the topology given in Figure 1. Note that the total number of links is 14. The assigned weights in this figure represent a possible configuration at time t, whereas the configuration represents a solution (i.e. a particle). A solution for this topology can be $(18, 1, 7, 15, 3, 17, 5, 14, 19, 13, 18, 4, 16, 16)$. This current solution is represented as

$$\mathbf{X}_i(t)=\{18_{AB}, 1_{AF}, 7_{BC}, 15_{BD}, 3_{CE}, 17_{CF},$$
$$5_{DA}, 14_{EA}, 19_{EG}, 13_{FB}, 18_{FD}, 4_{FG}, 16_{GB}, 16_{GD}\}.$$

Also assume that at time t, $\mathbf{V}_i(t)$ = $\{(19 \Leftrightarrow 18)_{AB}, (2 \Leftrightarrow 1)_{AF}, (4 \Leftrightarrow 7)_{BC}, (12 \Leftrightarrow 15)_{BD}, (4 \Leftrightarrow 3)_{CE}, (15 \Leftrightarrow 17)_{CF}, (6 \Leftrightarrow 5)_{DA}, (12 \Leftrightarrow 14)_{EA}, (13 \Leftrightarrow 19)_{EG}, (10 \Leftrightarrow 13)_{FB}, (11 \Leftrightarrow 18)_{FD}, (9 \Leftrightarrow 4)_{FG}, (17 \Leftrightarrow 16)_{GB}, (17 \Leftrightarrow 16)_{GD}\}$ where the symbol "$\Leftrightarrow$" represents a replacement of weights on the links. That is, the above solution, $\mathbf{X}_i(t)$, was obtained when weight 19 on link AB was replaced with a weight of 18, weight 2 on link AF

was replaced with a weight of 1, and so on. The solution $\mathbf{X}_i(t)$ is then updated in subsequent steps as discussed in the following subsections.

6.2 Velocity Update

The velocity of particle i is updated using

$$\mathbf{V}_i(t+1) = w \otimes \mathbf{V}_i(t) \oplus c_1 r_1(t) \otimes [\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)] \oplus c_2 r_2(t) \otimes [\mathbf{P}_g(t) \oslash \mathbf{X}_i(t)] \quad (11)$$

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

In Equation (11), the operator $\otimes$ is implemented as follows: The number of elements to be selected is determined as $\lfloor w \times |\mathbf{V}_i(t)| \rfloor$, where $0 < w < 1$. Then, the result is 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 a 'replacement' operator. For example, the weights in $\mathbf{X}_i(t)$ are replaced with the weights 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 randomly sampling $\lfloor c_1 r_1(t) \times |\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)| \rfloor$ elements from the set $\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)$, as follows:

$$c_1 r_1(t) \otimes [\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)] = \lfloor c_1 r_1(t) \times |\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)| \rfloor \quad (12)$$

where $|\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)|$ represents the cardinality of the set. The result of Equation (12) 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. 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). Further assume that the best goodness so far for particle i was generated by the following position as

$$\mathbf{P}_i(t) = \{18_{AB}, 12_{AF}, 7_{BC}, 15_{BD}, 3_{CE}, 16_{CF},$$
$$5_{DA}, 13_{EA}, 19_{EG}, 13_{FB}, 8_{FD}, 4_{FG}, 12_{GB}, 16_{GD}\}.$$

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

$$\mathbf{P}_g(t) = \{18_{AB}, 2_{AF}, 7_{BC}, 15_{BD}, 3_{CE}, 15_{CF},$$
$$5_{DA}, 13_{EA}, 19_{EG}, 13_{FB}, 9_{FD}, 4_{FG}, 1_{GB}, 16_{GD}\}.$$

The inertia weight, w, determines the number of replacements that will be randomly selected from $\mathbf{V}_i(t)$ (mentioned in Example 1 above). Since $w = 0.5$, and $|\mathbf{V}_i(t)|$ = 14, the number of randomly selected replacements is $0.5 \times |\mathbf{V}_i(t)|$ = 7. Thus, any seven replacements from the set $\mathbf{V}_i(t)$ can be taken randomly. Consider that those replacements are $\{(2 \Leftrightarrow 1)_{AF}, (4 \Leftrightarrow 7)_{BC}, (4 \Leftrightarrow 3)_{CE}, (6 \Leftrightarrow 5)_{DA}, (12 \Leftrightarrow 14)_{EA}, (13 \Leftrightarrow 19)_{EG}, (10 \Leftrightarrow 13)_{FB}\}$

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:

$\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)$ = $\{(18 \Leftrightarrow 18)_{AB}, (1 \Leftrightarrow 12)_{AF}, (7 \Leftrightarrow 7)_{BC}, (15 \Leftrightarrow 15)_{BD}, (3 \Leftrightarrow 3)_{CE}, (17 \Leftrightarrow 16)_{CF}, (5 \Leftrightarrow 5)_{DA}, (14 \Leftrightarrow 13)_{EA}, (19 \Leftrightarrow 19)_{EG}, (13 \Leftrightarrow 13)_{FB}, (18 \Leftrightarrow 8)_{FD}, (4 \Leftrightarrow 4)_{FG}, (16 \Leftrightarrow 12)_{GB}, (16 \Leftrightarrow 16)_{GD}\}$.

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 5, this implies that $0.5 \times 0.52 \otimes |\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)| = 1.3 = 1$. This

means that any one of the five elements in $\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)$ is randomly chosen. Assume that $c_1 \times r_1 \otimes (\mathbf{P}_i(t) \oslash \mathbf{X}_i(t))$ = $\{(18 \Leftrightarrow 8)_{FD}\}$.

Similarly,

$\mathbf{P}_g(t) \oslash \mathbf{X}_i(t)$ = $\{(18 \Leftrightarrow 18)_{AB}, (1 \Leftrightarrow 2)_{AF}, (7 \Leftrightarrow 7)_{BC}, (15 \Leftrightarrow 15)_{BD}, (3 \Leftrightarrow 3)_{CE}, (17 \Leftrightarrow 15)_{CF}, (5 \Leftrightarrow 5)_{DA}, (14 \Leftrightarrow 13)_{EA}, (19 \Leftrightarrow 19)_{EG}, (13 \Leftrightarrow 13)_{FB}, (18 \Leftrightarrow 9)_{FD}, (4 \Leftrightarrow 4)_{FG}, (16 \Leftrightarrow 1)_{GB}, (16 \Leftrightarrow 16)_{GD}\}$.

The cardinality of the above set is 5, since replacements involving new and old weights having the same value are ignored. Therefore, $0.5 \times 0.75 \otimes (\mathbf{P}_g(t) \oslash \mathbf{X}_i(t)) = 0.5 \times 0.75 \times 5$ = 1.3 = 1 replacement. Assume $\{(17 \Leftrightarrow 15)_{CF}\}$ is randomly chosen.

Substituting the above calculations in Equation (11) gives $\mathbf{V}_i(t+1)$ containing three elements, i.e.

$\mathbf{V}_i(t+1)$ = $\{(2 \Leftrightarrow 1)_{AF}, (4 \Leftrightarrow 7)_{BC}, (4 \Leftrightarrow 3)_{CE}, (6 \Leftrightarrow 5)_{DA}, (12 \Leftrightarrow 14)_{EA}, (13 \Leftrightarrow 19)_{EG}, (10 \Leftrightarrow 13)_{FB}, (18 \Leftrightarrow 8)_{FD}, (17 \Leftrightarrow 15)_{CF}\}$

Since V_{max} = 2, only two replacements from $\mathbf{V}_i(t+1)$ are randomly chosen. Assume that $((18 \Leftrightarrow 8)_{FD}$ and $(17 \Leftrightarrow 15)_{CF}$ are chosen. Then,

$\mathbf{V}_i(t+1)$ = $\{(18 \Leftrightarrow 8)_{FD}, (17 \Leftrightarrow 15)_{CF}\}$

6.3 Particle Position Update

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

$$\mathbf{X}_i(t+1) = \mathbf{X}_i(t) \biguplus \mathbf{V}_i(t+1) \tag{13}$$

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

Example 3: Consider Example 2, for which

$\mathbf{X}_i(t+1) = \mathbf{X}_i(t) \uplus \mathbf{V}_i(t+1)$ = $\{18_{AB}, 1_{AF}, 7_{BC}, 15_{BD}, 3_{CE}, 17_{CF}, 5_{DA}, 14_{EA}, 19_{EG}, 13_{FB}, 18_{FD}, 4_{FG}, 16_{GB}, 16_{GD}\} \uplus \{(18 \Leftrightarrow 8)_{FD}, (17 \Leftrightarrow 15)_{CF}\}$ = $\{18_{AB}, 1_{AF}, 7_{BC}, 15_{BD}, 3_{CE}, 15_{CF}, 5_{DA}, 14_{EA}, 19_{EG}, 13_{FB}, 8_{FD}, 4_{FG}, 16_{GB}, 16_{GD}\}$

Thus, in the new solution, weight 18 on link FD is replaced by 8 and weight 17 on link CF is replaced by 15.

6.4 Fitness Evaluation

Each iteration performs a few weight replacements. Because of weight changes, the routes within the network will change. The next step is to calculate the traffic on each link as a result of the new routes. Finally, the cost of the new solution is computed using Equation (10) as discussed in Section 5.

7 Fuzzy Evolutionary Particle Swarm Optimization Algorithm for Open Shortest Path First Weight Setting Problem

In addition to the fuzzy PSO algorithm for OSPFWS (described in Section 6), a hybrid variant of the fuzzy PSO using the simulated evolution (SimE) algorithm is presented in this section. This hybrid variant is referred to as fuzzy evolutionary PSO (FEPSO). Section 7.1 presents a brief discussion on SimE. This is followed by a discussion on FEPSO in Section 7.2.

7.1 Simulated Evolution Algorithm

Simulated evolution (SimE) is a search strategy proposed by Kling and Banerjee [9,53,54]. Throughout the search, SimE maintains a single solution which is perturbed to generate a

new solution. Each solution is comprised of a set of individuals, known as elements. SimE iteratively executes three steps:

- The **evaluation** step, which calculates the goodness of each element of the solution. The goodness of an element quantifies the nearness of the element with respect to its optimal value, and is a value in the range $[0, 1]$. The optimal value is problem specific and is determined theoretically or through some empirical analysis. A higher value of goodness indicates that the element is near to its optimal value.
- The **selection** step, in which a subset of elements are selected based on their goodness and removed from the current solution. The lower the goodness of a particular element, the higher its selection probability. A bias parameter B in the range [-1, 1] is used to control the number of elements selected. A negative value of B increases the number of elements selected in each iteration, thus favoring exploration. This may result in a high quality solution but at the expense of higher computational time. A positive value of B inflates the goodness of an element, thus reducing the number of elements being selected for reallocation. This may result in reduced execution time, but at the risk of premature convergence to sub-optimal (or local optimal) solution.
- The **allocation** step, in which the selected elements are allocated to new positions, with the intention of improving the existing solution. Each element selected in the selection step is removed from the solution and trial allocations are performed. The goodness of the solution resulting from each trial allocation is calculated, and the allocation which results in the highest goodness of the solution is accepted. This process of allocation and goodness calculations is repeated for each selected element. At the end of the allocation step, a new solution is obtained.

  Further details about the SimE algorithm can be found in [9,53,54].

7.2 Fuzzy Evolutionary Particle Swarm Optimization

Particles of the fuzzy PSO algorithm proposed in Section 6 perform weight replacements. These replacements involve replacing an old weights with a new weights on the links. Furthermore, the total number of performed replacements is limited by the parameter V_{max}. It is possible that, for a link i, a replacement may remove a weight (to be replaced with another weight) which might already be the optimum (or near-optimum) weight for that link. Note that this replacement is done 'blindly'. That is, the value of the new weight is chosen randomly. If these blind replacements continue for other links having optimum weights, then it will take a significant amount of time for the algorithm to converge. Rather than having a blind replacement, it would be more appropriate to replace a weight based on its quality. A weight with low quality will have a high probability of being removed from its current position, and vice versa. The question is how to measure the quality of a weight. This can be answered by incorporating the evaluation and selection phases of the SimE algorithm into the FPSO algorithm, as discussed below.

Recall from Section 7.1 that a solution in SimE is comprised of elements. For the OSPFWS problem, elements are the link weights, whose goodness need to be evaluated. In this paper, the function defined by Sqalli *et al.* [38] is employed to evaluate the goodness of a weight, as given below:

$$g_{ij} = \begin{cases} 1 - u_{ij} & \text{if } MU \leq 1 \\ 1 - u_{ij}/MU + u_{ij}/MU^2 & \text{if } MU > 1 \end{cases} \tag{14}$$

where u_{ij} represents the utilization on link connecting nodes i and j, and MU refers to the maximum utilization. The evaluation is performed for all current weights which are part of the set $\mathbf{V}_i(t+1)$ as defined by Equation (11).

Once the goodness of each existing weight in $\mathbf{V}_i(t+1)$ is evaluated, the selection phase chooses the weights that would be replaced with new weights. This selection is done probabilistically based on the quality of existing weights in $\mathbf{V}_i(t+1)$. A random number $Random$ in the range [0,1] is generated. If $Random \leq 1 - g_{ij} + B$, then the existing weight is selected for replacement, otherwise no replacement is done. In the above expression, g_{ij} refers to the goodness of current weight on the link connecting nodes i and j, and B is the selection bias. Figure 3 provides the pseudo-code of the selection function for FEPSO. The selection process is illustrated by the following example.

Example 4: In Example 2, $\mathbf{V}_i(t+1)$ was found as follows:

$\mathbf{V}_i(t+1)$ = $\{(2 \Leftrightarrow 1)_{AF}, (4 \Leftrightarrow 7)_{BC}, (4 \Leftrightarrow 3)_{CE}, (6 \Leftrightarrow 5)_{DA}, (12 \Leftrightarrow 14)_{EA}, (13 \Leftrightarrow 19)_{EG}, (10 \Leftrightarrow 13)_{FB}, (18 \Leftrightarrow 8)_{FD}, (17 \Leftrightarrow 15)_{CF}\}$

Since V_{max} = 2, only two replacements from $\mathbf{V}_i(t+1)$ were randomly chosen in FPSO. However, in FEPSO, the replacements will be done based on the goodness of weights. Assume that $(4 \Leftrightarrow 3)_{CE}$, $(10 \Leftrightarrow 13)_{FB}$ and $(17 \Leftrightarrow 15)_{CF}$ were selected based on the selection procedure. However, since V_{max} = 2, only two replacements will be selected randomly out of the three. So, a possible result could be

$\mathbf{V}_i(t+1)$ = $\{(10 \Leftrightarrow 13)_{FB}$ and $(17 \Leftrightarrow 15)_{CF}\}$

Although the proposed hybridization between PSO and SimE seems promising in generating high quality results, the approach also has some negative aspects. One major complexity is associated with tuning of another parameter, namely, *bias B*, on top of tuning of various parameters associated with the PSO algorithm. This adds extra effort and time to find the best combination out of many possible combinations of all design parameters. Another issue is that the complexity of the code increases, thus increasing the execution time. A single iteration of FEPSO will take more time than a single iteration of FPSO. In a broader perspective and with consideration of applying the proposed hybrid algorithm to other optimization

Function *Selection (B)*;

/* B: Selection Bias; */

Get the set of possible replacements $V_i(t+1)$ from Equation (1)

For all the current weights in the set $V_i(t+1)$ **do**

Calculate the goodness g_{ij} of weight on link between nodes i and j using Equation (2)

If $Random \leq 1 - g_{ij} + B$ **Then**

Allow the replacement of the old weight with new weight;

Else

Don't allow the replacement of the old weight with new weight;

End If

End For

End *Selection*;

Fig. 3 Weight replacement function of FEPSO

problems, it can be argued that the proposed hybridization will allow faster convergence of FEPSO to an optimal or sub-optimal solution as compared to FPSO. However, this is not always guaranteed and can only be established after thorough experimentation and analysis.

8 Experimental methodology

This paper uses test cases from [2] which have been used by many other researchers as discussed in Section 2. Table 1 summarizes the characteristics of the test cases. For each test case, the table lists its network type, the number of nodes (N), and the number of arcs or edges (a). The *2-level hierarchical networks* are generated using the GT-ITM generator [55], based on the model of Calvert [56] and Zegura [57]. In hierarchical networks, local access arcs have capacities equal to 200, while long distance arcs have capacities equal to 1000. In *Random networks* and *Waxman networks*, capacities are set at 1000 for all arcs. Fortz and

Thorup generated the demands to force some nodes to be more active senders or receivers than others, thus modelling *hot spots* on the network. More specifically, higher demands were assigned to closely located node pairs. Further details can be found in [8].

Experiments were done with different combinations of PSO parameters for each test case. Thirty independent runs were executed for each parameter setup, and the average of the best solutions found in each run was reported, with the associated standard deviation. Furthermore, results were validated for statistical significance through non-parametric testing. For this purpose, the Wilcoxon's rank-sum test was used with confidence level set at 95%. After experimenting with different values, it was found that 100 iterations were reasonable to observe the trends. Therefore, each run was executed for 100 iterations.

Table 1 Test cases for the OSPFWS problem. N = number of nodes, a = number of arcs

Test Code	**Network type**	**N**	**a**
h100N280a	2-level hierarchical graph	100	280
h100N360a	2-level hierarchical graph	100	360
h50N148a	2-level hierarchical graph	50	148
h50N212a	2-level hierarchical graph	50	212
r100N403a	Random graph	100	403
r100N503a	Random graph	100	503
r50N228a	Random graph	50	228
r50N245a	Random graph	50	245
w100N391a	Waxman graph	100	391
w100N476a	Waxman graph	100	476
w50N169a	Waxman graph	50	169
w50N230a	Waxman graph	50	230

Table 2 PSO parameter settings used in the experiments

Parameter Name	**Parameter Values**
swarm size	20, 40, 60, 80, 100
V_{max}	5, 10, 15, 20
w	0.3, 0.5, 0.72, 0.85, 0.99
c_1, c_2	0.7 and 1.4, 1.4 and 0.7, 1.49 and 1.49, 2 and 2. 0.5 and 3, 3 and 0.5

9 Results and Discussion

The proposed PSO algorithm was evaluated with respect to all the PSO parameters. These parameters are the swarm size, acceleration coefficients c_1 and c_2, inertia weight w, and velocity clamping V_{max}. Table 2 lists all the parameter combinations used. The following parameters were used as default: swarm size $= 40$, $V_{max} = 15$, $w = 0.72$, and c_1 = c_2 = 1.49. The values $c_1 = c_2$ = 1.49 (along with $w = 0.72$) were specifically selected, since they are frequently used in the literature due to the fact that they enhance the probability of convergence [58].

9.1 Effect of Swarm Size

The effect of swarm size was investigated with 20, 40, 60, 80, and 100 particles, as listed in Table 2. Other parameter values were kept at the defaults. Figures 4 and 5 provide a graphical representation of the effect of varying the swarm size on the quality of solutions obtained for test cases with 50 nodes and 100 nodes, respectively (detailed results are presented in Appendix B). It is observed from the figures that for all test cases, increasing the number of particles enhanced the quality of solution. More specifically, the highest average overall goodness was obtained with the highest swarm size consisting of 100 particles, while the lowest average overall goodness was obtained with the minimum value of swarm size, i.e.

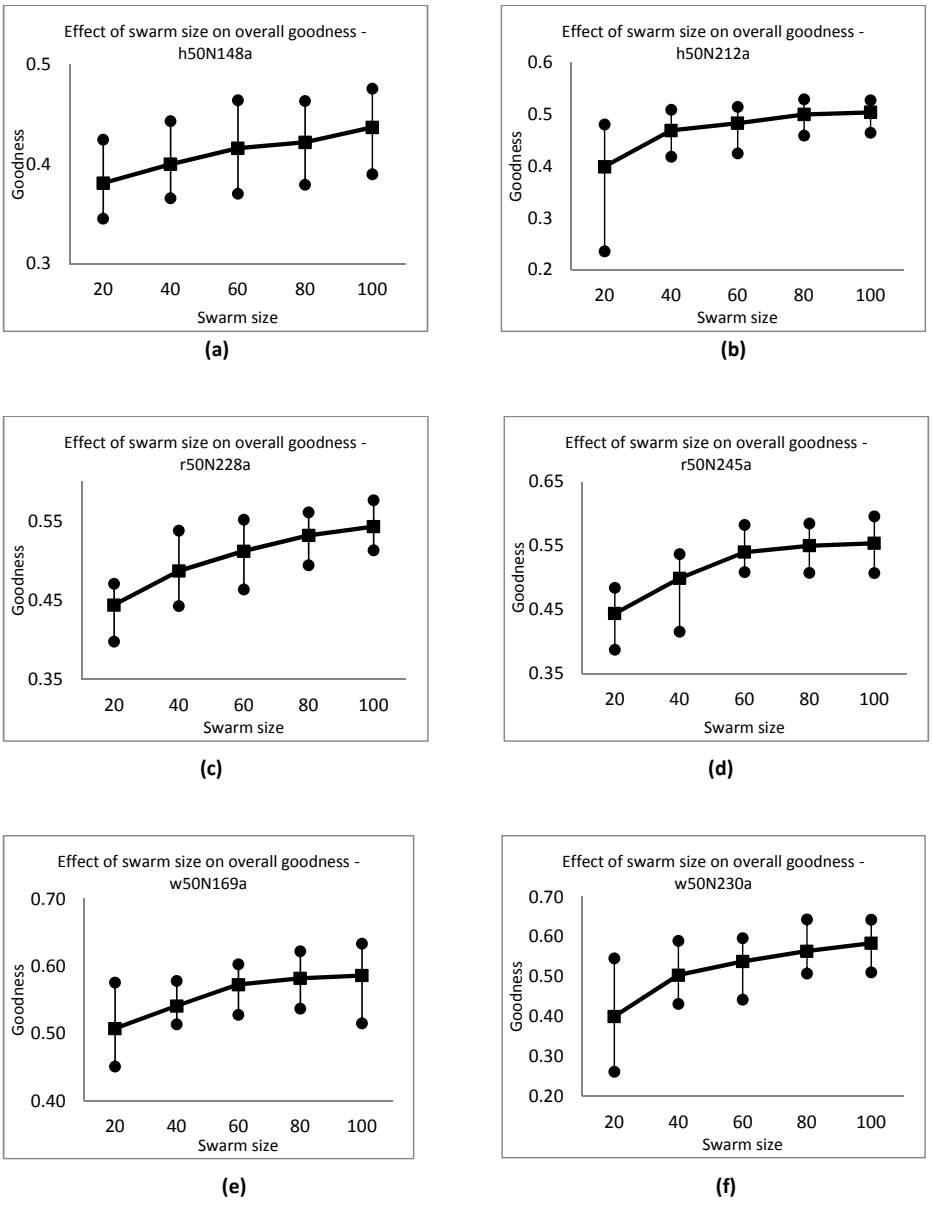

Fig. 4 Effect of swarm size on overall goodness for test cases with 50 nodes: (a) h50N148a (b) h50N212a (c) r50N228a (d) r50N245a (e) w50N169a (f) w50N230a

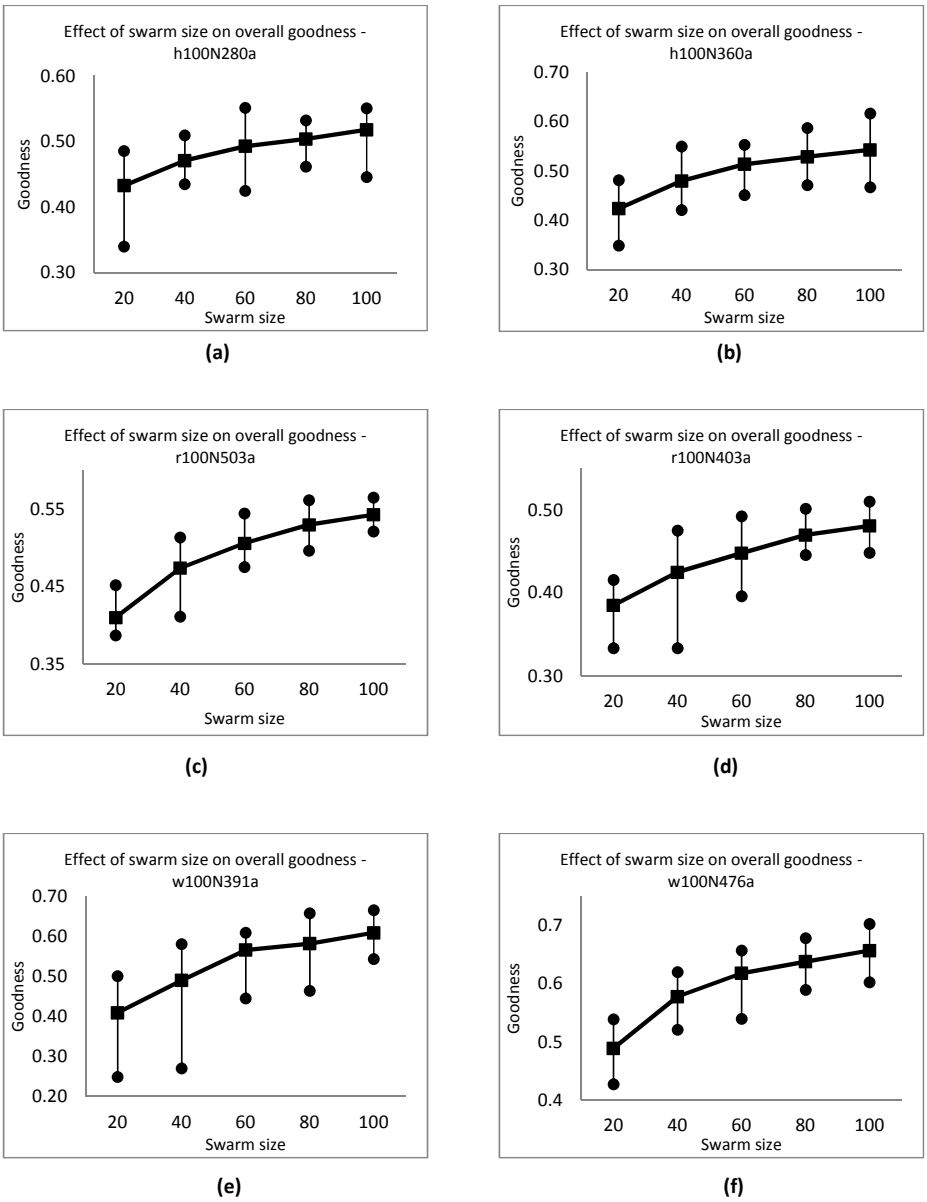

Fig. 5 Effect of swarm size on overall goodness for test cases with 100 nodes: (a) h100N280a (b) h100N360a (c) r100N503a (d) r100N403a (e) w100N391a (f) w100N476a

20 particles. Furthermore, the figures show a logarithmic decrease in the gains in quality with increase in swarm size. A validation with Wilcoxon's test with 95% significance level for the average overall goodness obtained with different swarm sizes was also performed for the results reported in Tables 14 to 25 (refer to Appendix B). The hypothesis testing was done to check whether the average goodness value obtained with 100 particles was statistically significantly better than those obtained with other swarm sizes. The results confirmed that a swarm size of 100 gave the best results for all test cases.

Another important observation from Tables 14 to 25 is that, for most test cases, swarms of 80 and 100 particles resulted in no significant difference with respect to solution quality. This happened for the test cases h100N280a, h100N360a, h50N212a, r100N503a, r50N228a, r50N245a, w50N169a, and w50N230a. The test case w50N169a was an exception, where swarm sizes 60, 80, and 100 particles resulted in the same quality of solutions. Therefore, the smaller swarm size was preferred over larger swarm size due to lower computational cost.

Diversity is defined as a measure of the average distance of each particle from the center of the mass. Diversity is calculated at each iteration during the execution of the algorithm [59]. The effect of increase in swarm size on diversity was also studied. The purpose was to observe whether bigger swarm sizes reduced the possibility of getting trapped in local minima (preventing premature convergence), thus resulting in solutions of higher quality. As an example, Figure 6 shows the diversity plots for different number of particles for the test case w100N476a. The figure suggests that the algorithm did not converge immediately after initialization for all the swarms. Diversity increased until around iteration number 50. The reduction in diversity is seen when the algorithm started converging. Swarms with 20 particles maintained diversity at a higher level compared to other swarm sizes, and quickly converged at around iteration 225. This is followed by swarms with 40 and 60 particles,

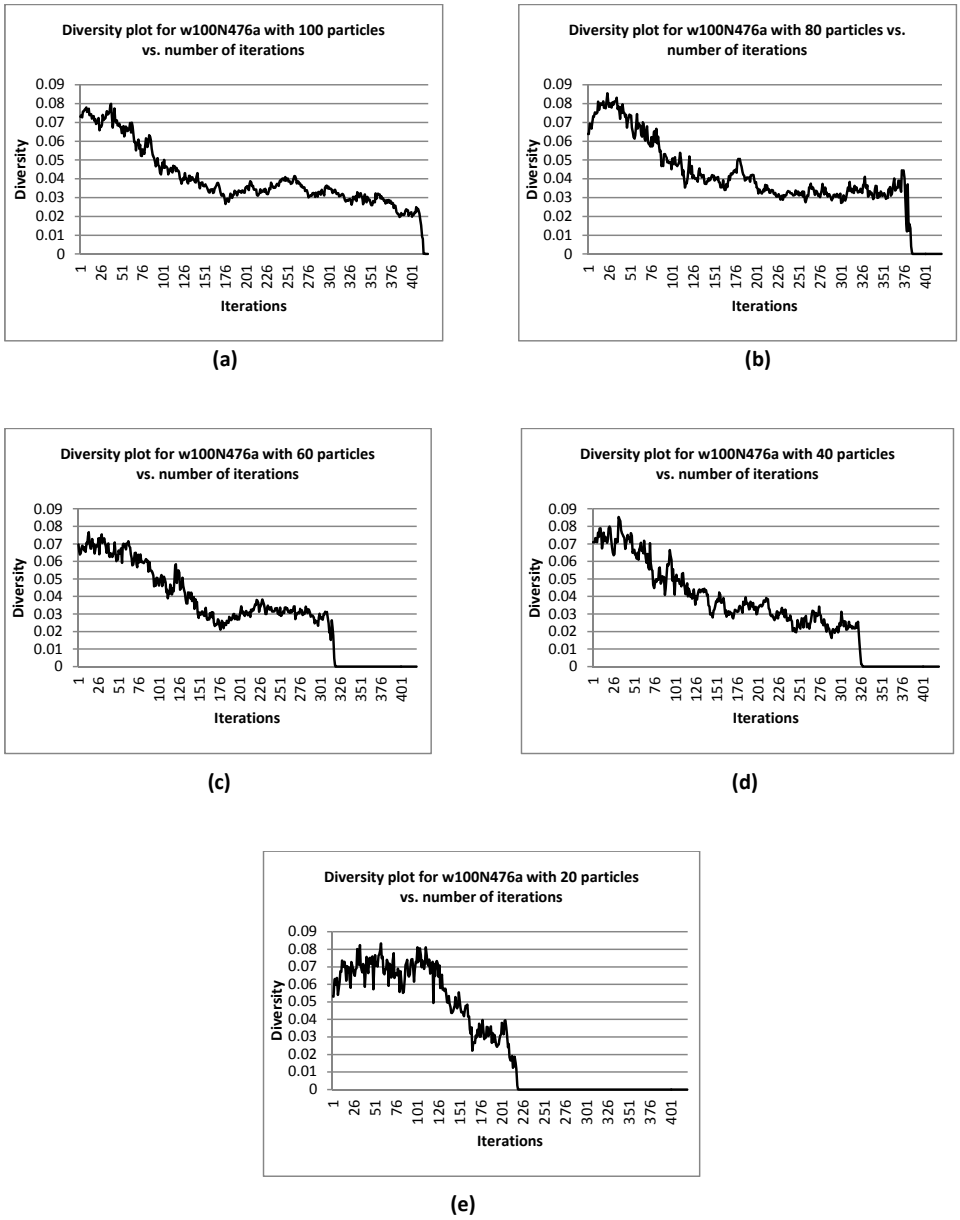

Fig. 6 Diversity plots for *w100N476a* using (a) 100 particles (b) 80 particles (c) 60 particles (d) 40 particles and (e) 20 particles

which converged nearly at the same time, at around iteration number 326. Followed by this, swarm with 80 particles converged at iteration 380. Finally, the swarm with 100 particles converged in the last, after 400 iterations.

9.2 Effect of Acceleration Coefficients

The impact of the acceleration coefficients on the algorithm's performance was also evaluated. Table 3 gives the average overall goodness for different combinations of acceleration coefficients, as given in Table 2. Other algorithm parameters were kept as their defaults.

As observed in Table 3, a dominant trend was that the best results were obtained when the value of c_1 was much higher than that of c_2. That is, for seven out of twelve test cases, the best overall goodness was obtained when c_1 = 3.0 and c_2 = 0.5. Furthermore, there were two other test cases (r100N403a and w100N391a) where the best overall goodness was obtained with c_1 = 1.4 and c_2 = 0.7. These results indicate that in general, the algorithm resulted in highest overall goodness values when the cognitive component dominated the social component. There were only three test cases (h100N280a, h100N360a and w50N169a) which deviated from the above trend.

The above observations are further supported by the results given in Table 4 which gives the percentage improvements in terms of the best and worst overall goodness values. The results show that the level of improvements achieved ranged between 2.61% and 11.74%. In most instances, the average overall goodness was around 4% or above. Statistical validation with the Wilcoxon's test proved that in a majority of the cases, the improvements were statistically significant (highlighted in boldface). It is clear from these results that for eight test cases, best overall goodness values were obtained when the value of c_1 was greater than c_2. There were three exceptions, namely, h100N280a, w50N169a, and h100N360 (highlighted

in asterisk) which deviated from the above trend. Furthermore, improvements in one test case (r50N245a) turned out to be statistically insignificant. In view of the above results and discussion, it can be fairly concluded that higher quality results produced by PSO were governed by the cognitive component.

Table 3 Effect of different acceleration coefficients combinations on overall goodness for different test cases. Best overall goodness for each test case is in boldface

Test Case	c_1 = 1.49 c_2 = 1.49 (Set 1)	c_1 = 0.7 c_2 = 1.4 (Set 2)	c_1 = 1.4 c_2 = 0.7 (Set 3)	c_1 = 2.0 c_2 = 2.0 (Set 4)	c_1 = 0.5 c_2 = 3.0 (Set 5)	c_1 = 3.0 c_2 = 0.5 (Set 6)
h100N280a	0.471±0.018	**0.494**±0.029	0.473±0.027	0.486±0.023	0.464±0.022	0.468±0.031
h100N360a	0.480±0.034	0.470±0.052	0.478±0.029	0.480±0.032	**0.493**±0.044	0.484±0.021
h50N148a	0.400±0.019	0.401±0.019	0.409±0.023	0.404±0.017	0.403±0.033	**0.427**±0.022
h50N212a	0.469±0.022	0.445±0.038	0.460±0.051	0.450±0.052	0.484±0.011	**0.494**±0.021
r100N403a	0.425±0.031	0.432±0.02	**0.437**±0.016	0.430±0.019	0.410±0.017	0.418±0.052
r100N503a	0.474±0.029	0.467±0.024	0.474±0.021	0.473±0.021	0.485±0.019	**0.486**±0.027
r50N228a	0.487±0.022	0.485±0.02	0.490±0.023	0.492±0.026	0.488±0.012	**0.504**±0.035
r50N245a	0.499±0.03	0.509±0.023	0.507±0.022	0.501±0.026	0.500±0.022	**0.512**±0.022
w100N391a	0.490±0.063	0.512±0.048	**0.514**±0.07	0.507±0.086	0.500±0.044	0.482±0.118
w100N476a	0.578±0.026	0.572±0.03	0.568±0.025	0.570±0.022	0.567±0.025	**0.595**±0.010
w50N169a	0.541±0.019	0.551±0.03	0.536±0.029	**0.552**±0.028	0.523±0.028	0.530±0.010
w50N230a	0.504±0.038	0.501±0.043	0.477±0.079	0.513±0.042	0.460±0.087	**0.514**±0.049

9.3 Effect of Inertia Weight

The effect of the inertia weight was assessed with respect to the five values listed in Table 2. Other algorithm parameters were kept at their defaults. Table 5 gives the average overall

Table 4 Results for best and worst overall goodness and their corresponding values of c_1 and c_2. Statistically significant improvements are given in boldface.

Test case	Best overall goodness	c_1, c_2	Worst overall goodness	c_1, c_2	% difference
h100N280a	0.494	0.7, 1.4	0.464	0.5, 3	6.47 *
h100N360a	0.493	0.5, 3	0.480	0.7, 1.4	4.89 *
h50N148a	0.427	3, 0.5	0.400	1.49, 1.49	**6.75**
h50N212a	0.494	3, 0.5	0.445	0.7, 1.4	**11.01**
r100N403a	0.437	1.4, 0.7	0.410	0.5, 3	**6.59**
r100N503a	0.486	3, 0.5	0.467	0.7, 1.4	**4.07**
r50N228a	0.504	3, 0.5	0.485	0.7, 1.4	**3.92**
r50N245a	0.512	3, 0.5	0.499	1.49, 1.49	2.61
w100N391a	0.514	1.4, 0.7	0.482	3, 0.5	**6.64**
w100N476a	0.595	3, 0.5	0.567	0.5, 3	**4.94**
w50N169a	0.552	2, 2	0.523	0.5, 3	5.54 *
w50N230a	0.514	3, 0.5	0.460	0.5, 3	**11.74**

goodness for the different values of the inertia weight. It is observed from this table that, in general, higher values of the inertia weight (w = 0.85 and w = 0.99) tend to show better results than the tried smaller values.

In order to validate the above observations, statistical testing was done and results are shown in Table 6. This table shows the percentage improvements obtained when results with two different inertia weights were compared. Although the results show improvements in terms of percentages, statistical testing revealed that in general, improvements were statistically insignificant. Only few improvements were significant which are highlighted in boldface. Based on these observations, it can be confidently claimed that with respect to

the five values tried, the inertia weight did not have a notable impact on the output results produced by the algorithm.

Table 5 Average overall goodness values achieved with different inertia weights. Best overall goodness values are given in boldface.

Test Case	w = 0.3	w = 0.5	w = 0.72	w = 0.85	w = 0.99
h100N280a	0.469±0.025	0.469±0.029	0.471±0.018	0.477±0.013	**0.479** ±0.024
h100N360a	0.467±0.051	0.468±0.041	0.48±0.034	**0.495**±0.026	0.462±0.019
h50N148a	**0.406**±0.027	0.391±0.025	0.400±0.019	0.373±0.027	0.396±0.035
h50N212a	0.466±0.028	0.447±0.03	0.469±0.022	**0.483**±0.014	0.461±0.014
r100N403a	**0.431**±0.025	0.421±0.028	0.425±0.031	0.423±0.008	0.409±0.025
r100N503a	0.465±0.026	0.465±0.021	**0.474**±0.029	0.473±0.011	0.462±0.041
r50N228a	0.495±0.021	0.491±0.022	0.487±0.022	0.495±0.034	**0.502**±0.023
r50N245a	0.500±0.02	0.509±0.022	0.499±0.03	**0.524**±0.027	0.504±0.026
w100N391a	0.474±0.071	0.509±0.052	0.490±0.063	**0.513**±0.023	0.417±0.137
w100N476a	**0.586**±0.027	0.57±0.023	0.578±0.026	0.572±0.055	0.584±0.048
w50N169a	0.542±0.027	0.550±0.023	0.541±0.019	0.552±0.033	**0.559**±0.023
w50N230a	0.501±0.033	0.501±0.043	0.504±0.038	0.518±0.046	**0.526**±0.024

9.4 Effect of Velocity Clamping

The effect of velocity clamping was also investigated. Apart from the default value of V_{max} = 15, other values of velocity clamping 5, 10, and 20, were tried as listed in Table 2. Other algorithm parameters were kept as defaults. The inspiration for taking the aforementioned values of V_{max} comes from the concept of mutation rates in genetic algorithms. Note that the function of V_{max} in PSO and mutation in GA is somewhat similar; both parameters control the level of perturbation in the solution. Although the mutation rate is problem spe-

Table 6 Comparison of different inertia weights given in Table 5 in terms of percentage differences. Statistically significant differences are highlighted in boldface.

Test Case	0.3 vs 0.5	0.3 vs 0.72	0.3 vs 0.85	0.3 vs 0.99	0.5 vs 0.72	0.5 vs 0.85	0.5 vs 0.99	0.72 vs 0.85	0.72 vs 0.99	0.85 vs 0.99
h100N280a	0.00	0.43	1.71	2.13	0.43	1.71	2.13	1.27	1.70	0.42
h100N360a	-0.21	2.78	**6.00**	-1.07	2.56	**5.77**	-1.28	3.13	-3.75	**-6.67**
h50N148a	3.69	-1.48	**-8.13**	-2.46	2.30	**-4.60**	1.28	**-6.75**	-1.00	**6.17**
h50N212a	**4.08**	0.64	3.65	-1.07	**4.92**	**8.05**	3.13	2.99	-1.71	**-4.55**
r100N403a	2.32	-1.39	-1.86	**-5.10**	0.95	0.48	-2.85	-0.47	-3.76	-3.31
r100N503a	0.00	1.94	1.72	-0.65	1.94	1.72	-0.65	-0.21	-2.53	-2.33
r50N228a	0.81	-1.62	0.00	1.41	-0.81	0.81	2.24	1.64	3.08	1.41
r50N245a	-1.80	-0.20	**4.80**	0.80	-1.96	2.95	-0.98	**5.01**	1.00	-3.82
w100N391a	**-7.38**	3.38	**8.23**	**-12.03**	-3.73	0.79	**-18.07**	**4.69**	**-14.90**	**-18.71**
w100N476a	2.73	-1.37	-2.39	-0.34	1.40	0.35	2.46	-1.04	1.04	2.10
w50N169a	-1.48	-0.18	1.85	3.14	-1.64	0.36	1.64	2.03	3.33	1.27
w50N230a	0.00	0.60	3.39	**4.99**	0.60	3.39	**4.99**	2.78	**4.37**	1.54

cific, a number of studies [59–63] have used the mutation rate of up to 20%. Therefore, the motivation for choosing the given range of V_{max} is the above observation. For the problem in hand, since the size of the solution string varies between 148 and 503 (corresponding to the number of edges on which the weights are varied), the values of V_{max} ranging from 5 to 20 correspond to perturbation rates of 1% to around 12%.

Table 7 shows the average overall goodness for the four values of V_{max} investigated. It is observed from the table that V_{max} = 5 produced the highest values of average overall goodness for almost all test cases, with the exception of r100N403a, where V_{max} = 10 produced the highest value of average overall goodness. Another deviation from the trend

was the case w100N391a where both V_{max} = 5 and V_{max} = 10 produced the same level of average overall goodness.

The best performance of V_{max} = 5 was further confirmed by statistical testing which showed that the results produced by V_{max} = 5 were indeed significant for all test cases when compared with V_{max} = 15 and V_{max} = 20, as depicted in Table 8. Furthermore, when compared with V_{max} = 10, the results produced by V_{max} = 5 were statistically significant for 9 out of the 12 test cases. The above observations and analysis clearly indicate that V_{max} = 5 resulted in the highest quality of solutions compared to the other values of V_{max} tested.

It is obvious from the above discussion and analysis that V_{max} had a significant impact on the quality of final solutions produced by the proposed PSO algorithm. The results also indicate that better overall goodness values were obtained when velocity clamping was kept low. This can be attributed to the fact that, with larger values of V_{max}, the algorithm was biased towards exploration which rather resulted in more randomized search. A lower value of V_{max} was therefore able to better balance exploration and exploitation.

9.5 Comparison of Fuzzy Particle Swarm Optimization and Fuzzy Evolutionary Particle Swarm Optimization

Table 9 summarizes the comparison of the proposed fuzzy PSO and the fuzzy evolutionary PSO. The table shows the results of the best parameter combination for FPSO and the corresponding results for FEPSO for the same parameter combination. Thirty runs were executed for each test cases and results were statistically validated through Wilcoxon's rank-sum test. The execution time (not the number of iterations) for both versions was also kept the same. It is clearly observed from the table that the improvements achieved by FESPO were statistically significant for all test cases, with the exception of h100N280a (for which FEPSO

Table 7 Analysis of velocity clamping. Best overall goodness is in boldface.

Test Case	V_{max}=15 Set (1)	V_{max}=5 Set (2)	V_{max}=10 Set (3)	V_{max}=20 Set (4)
h100N280a	0.471±0.018	**0.531**±0.018	0.489±0.028	0.454±0.021
h100N360a	0.48±0.034	**0.539**±0.025	0.482±0.051	0.45±0.039
h50N148a	0.4±0.019	**0.437**±0.021	0.414±0.022	0.387±0.023
h50N212a	0.469±0.022	**0.502**±0.02	0.489±0.019	0.436±0.038
r100N403a	0.425±0.031	0.447±0.025	**0.45**±0.014	0.415±0.017
r100N503a	0.474±0.029	**0.535**±0.023	0.51±0.023	0.462±0.02
r50N228a	0.487±0.022	**0.543**±0.019	0.517±0.021	0.473±0.023
r50N245a	0.499±0.03	**0.557**±0.019	0.53±0.024	0.485±0.023
w100N391a	0.49±0.063	**0.538**±0.103	**0.538**±0.05	0.456±0.083
w100N476a	0.578±0.026	**0.641**±0.023	0.615±0.03	0.544±0.053
w50N169a	0.541±0.019	**0.595**±0.021	0.57±0.033	0.519±0.026
w50N230a	0.504±0.038	**0.591**±0.029	0.557±0.028	0.478±0.041

had a slightly inferior performance with degradation of 0.91% in the average overall goodness). However, statistical analysis suggested that this deteriorated result was not significant. Therefore, it can be confidently claimed that FEPSO outperformed FPSO in terms of quality of the average overall goodness.

The superior performance of FEPSO can be attributed to its design. Recall from Section 7.2 that an existing solution is perturbed by performing moves as governed by Equation (11). These moves result in randomly replacing existing weights on links with new weights. It is quite possible that some of these moves may result in removing a near-optimum (or even optimum) weight from a certain link and introducing an unsuitable weight. In order to avoid this from happening, the concept of "intelligent" move was introduced in FEPSO, which

Table 8 Comparison of different values of velocity clamping. Significant differences are highlighted in boldface.

Test Case	V_{max}=15 vs V_{max}=5	V_{max}=15 vs V_{max}=10	V_{max}=15 vs V_{max}=20	V_{max}=5 vs V_{max}=10	V_{max}=5 vs V_{max}=20	V_{max}=10 vs V_{max}=20
h100N280a	**-11.3**	-3.68	**3.74**	**8.59**	**16.96**	**7.71**
h100N360a	**-10.95**	-0.41	**6.67**	**11.83**	**19.78**	**7.11**
h50N148a	**-8.47**	**-3.38**	3.36	**5.56**	**12.92**	**6.98**
h50N212a	**-6.57**	**-4.09**	**7.57**	2.66	**15.14**	**12.16**
r100N403a	**-4.92**	**-5.56**	2.41	-0.67	**7.71**	**8.43**
r100N503a	**-11.4**	**-7.06**	2.6	**4.9**	**15.8**	**10.39**
r50N228a	**-10.31**	**-5.8**	2.96	**5.03**	**14.8**	**9.3**
r50N245a	**-10.41**	**-5.85**	2.89	**5.09**	**14.85**	**9.28**
w100N391a	**-8.92**	**-8.92**	**7.46**	0	**17.98**	**17.98**
w100N476a	**-9.83**	**-6.02**	**6.25**	**4.23**	**17.83**	**13.05**
w50N169a	**-9.08**	**-5.09**	**4.24**	**4.39**	**14.64**	**9.83**
w50N230a	**-14.72**	**-9.52**	**5.44**	**6.1**	**23.64**	**16.53**

allows the algorithm to converge in less amount of time, or alternatively speaking, producing higher quality results when executed for the same amount of time as that of FPSO.

10 Comparison of Fuzzy Evolutionary Particle Swarm Optimization with other algorithms

Since fuzzy evolutionary PSO algorithm (FEPSO) performed better than fuzzy PSO (FPSO) algorithm, it was compared with other iterative heuristics, namely, Pareto-dominance PSO (PDPSO) [64], PSO with weighted aggregation (WAPSO) [65], non-dominated sorting genetic algorithm II (NSGA-II)[66,42], simulated evolution (SimE)[40,42], and simulated

Table 9 Comparison of fuzzy PSO and fuzzy evolutionary PSO. Significant differences are highlighted in bold. NoP = number of particles, % Imp = percentage improvement. Runtime is in seconds.

Test Case	NoP	C_1	C_2	W	V_{max}	Run Time	FPSO	FEPSO	% Imp
h100N280a	40	1.49	1.49	0.72	5	12460.7	0.531±0.018	0.526±0.015	-0.91
h100N360a	100	1.49	1.49	0.72	15	30679.8	0.543±0.036	0.605±0.012	**11.40**
h50N148a	40	1.49	1.49	0.72	5	808.9	0.437±0.021	0.469±0.019	**7.22**
h50N212a	100	1.49	1.49	0.72	15	2291.2	0.504±0.015	0.528±0.013	**4.94**
r100N403a	100	1.49	1.49	0.72	15	62095.1	0.481±0.017	0.595±0.011	**23.73**
r100N503a	100	1.49	1.49	0.72	15	78408.6	0.543±0.013	0.710±0.012	**30.83**
r50N228a	100	1.49	1.49	0.72	15	3112.4	0.543±0.019	0.610±0.016	**12.27**
r50N245a	40	1.49	1.49	0.72	5	1586.8	0.557±0.019	0.644±0.014	**15.67**
w100N391a	100	1.49	1.49	0.72	15	48083.9	0.609±0.029	0.725±0.010	**18.94**
w100N476a	100	1.49	1.49	0.72	15	71213.2	0.657±0.025	0.757±0.011	**15.24**
w50N169a	40	1.49	1.49	0.72	5	1084.5	0.595±0.021	0.640±0.012	**7.47**
w50N230a	40	1.49	1.49	0.72	5	1653.0	0.591±0.029	0.711±0.022	**20.48**

annealing (SA) [39,42]. PDPSO and WAPSO were adapted for the underlying problem, whereas NSGA-II, SimE, and SA have already been applied to the same problem by Mohiuddin *et al.* [42]. Details of implementation and comparative analysis of NSGA-II, SimE, and SA can be found in Mohiuddin *et al.* [42]. Thirty runs were made for each test cases for each algorithm, and average results and standard deviations were reported. All algorithms were run for the same amount of time for fair comparisons.

Tables 10, 11, and 12 present the results obtained for FEPSO, PDPSO, WAPSO, NSGA-II, SimE, and SA for the three objectives, respectively. Since the multi-objective assessment approach for the algorithms is not the same, the overall objective function that shows the

combined effect of all three objectives cannot be directly used for comparison. Therefore, each objective was evaluated individually. Table 10 indicates that for the maximum utilization objective, FEPSO obtained the best results for four test cases (h100N360a, r100N503a, r50N245a, w50N230a). For two test cases (r100N403a and r50N228a), both FEPSO and SimE produced the best results. For four cases (h100N280a, h40N212a, w100N391a, and w100N476a), SimE generated the best (minimum) values. For two test cases (h50N148a and w50N169a), SA obtained the best results. As for the objective of number of congested links, the results in Table 11 indicate that FEPSO obtained the best results for nine test cases. The exception was three test cases of h100N280a, h50N148a, and w50N169a where SimE, SA, and NSGA-II were able to achieve the best values, respectively. Finally for the objective of number of unused links, the results in Table 10 indicate that FEPSO was able to achieve op-

Table 10 Comparison of Maximum Utilization (MU) achieved by FEPSO, PDPSO, WAPSO, NSGA-II, SimE, and SA. Best (minimum) values are shown in bold.

Test Case	FEPSO	PDPSO	WAPSO	NSGA-II	SimE	SA
h100N280a	1.44±0.06	1.42±0.07	1.44±0.10	1.42±0.07	**1.41**±0.06	1.50±0.35
h100N360a	**1.69**±0.08	1.86±0.07	1.82±0.12	1.84±0.08	1.72±0.09	2.01±0.53
h50N148a	1.54±0.09	1.71±0.09	1.67±0.13	2.90±1.12	1.62±0.12	**1.51**±0.1
h50N212a	1.71±0.05	1.76±0.05	1.80±0.05	1.71±0.08	**1.67**±0.1	1.68±0.08
r100N403a	**1.86**±0.07	2.35±0.18	2.32±0.12	4.13±1.93	**1.86**±0.07	2.72±0.58
r100N503a	**1.95**±0.08	3.40±0.24	2.85±0.25	3.48±0.21	2.19±0.17	3.58±0.42
r50N228a	**1.80**±0.11	2.09±0.14	2.04±0.21	2.05±0.16	**1.80**±0.12	2.02±0.13
r50N245a	**2.44**±0.19	2.99±0.28	2.66±0.21	2.90±0.29	2.60±0.19	2.83±0.24
w100N391a	1.49±0.02	1.68±0.13	1.69±0.13	1.65±0.11	**1.42**±0.03	1.75±0.71
w100N476a	1.54±0.07	1.95±0.14	1.94±0.15	2.17±0.2	**1.46**±0.07	2.24±0.22
w50N169a	1.44±0.08	1.60±0.11	1.67±0.06	1.43±0.07	1.44±0.08	**1.41**±0.09
w50N230a	**1.36**±0.09	1.71±0.16	1.61±0.12	1.65±0.10	1.44±0.09	1.55±0.15

Table 11 Comparison of Number of Congested Links (NOC) achieved by FEPSO, PDPSO, WAPSO, NSGA-II, SimE, and SA. Best (minimum) values are shown in bold.

Test Case	FEPSO	PDPSO	WAPSO	NSGA-II	SimE	SA
h100N280a	9.00±1.26	9.00±1.20	9.00±1.49	8.65±1.04	**8.50**±1.43	8.93±1.39
h100N360a	**12.80**±1.99	16.60±3.27	15.90±2.38	16.80±2.35	16.13±2.24	21.03±6.30
h50N148a	9.80±1.15	11.10±2.08	9.20±2.20	16.45±3.99	10.37±2.03	**8.4**±1.67
h50N212a	**4.70**±0.66	5.20±0.79	4.80±0.42	5.80±0.83	4.93±0.74	5.17±0.79
r100N403a	**32.95**±2.76	54.50±5.82	58.80±3.16	72.25±19.06	44.73±2.49	62.6±6.58
r100N503a	**31.45**±2.95	73.80±5.05	69.40±4.74	73.75±3.18	52.80±2.81	82.33±26.21
r50N228a	**16.65**±1.57	22.10±2.08	24.60±2.37	19.95±2.11	19.77±1.30	22.03±2.25
r50N245a	**20.25**±1.48	27.50±2.32	28.30±2.00	26.05±2.54	26.20±1.97	28.77±2.62
w100N391a	**3.25**±1.12	16.50±3.60	12.50±4.62	14.15±3.72	7.17±2.69	42.1±19.92
w100N476a	**11.25**±1.89	23.30±3.13	24.30±3.06	30.00±4.12	17.07±2.36	41.7±15.65
w50N169a	7.70±0.92	9.70±1.16	8.80±2.04	**7.65**±1.35	8.37±1.19	8.8±1.81
w50N230a	**6.00**±1.21	13.40±3.66	11.70±2.06	11.25±2.10	9.13±1.50	14.4±9.08

timum values (i.e., no link was left unused) in all test cases, with the exception of r50N245a. However, it should also be noted that there are many instances where other algorithms also achieved the optimum levels.

Although the discussion above provides a fair picture of the performance of FEPSO with respect to individual objectives, an overall view of FEPSO‘s performance is desired. Table 13 accumulates the results with regard to the three objectives and displays the best and worst achievers for each objective. The results given in Table 13 are based on results of Tables 10, 11, and 12. The table signifies that in five cases (h100N360a, r100N403a, r100N503a, r50N228a, and w50N230a), FEPSO achieved the best results for all three objectives, while there are four cases (h50N212a, r50N245a, w100N391a, and w100N476a) where FEPSO

Table 12 Comparison of Number of Unused Links (NUL) achieved by FEPSO, PDPSO, WAPSO, NSGA-II, SimE, and SA. Best (minimum) values are shown in bold.

Test Case	FEPSO	PDPSO	WAPSO	NSGA-II	SimE	SA
h100N280a	0.00±0.00	0.00±0.00	0.10±0.32	0.00±0.00	1.23±0.90	0.7±3.31
h100N360a	0.00±0.00	0.10±0.32	0.00±0.00	0.20±0.52	0.93±1.01	0.2±0.48
h50N148a	0.00±0.00	0.00 ±0.00	0.00±0.00	0.30±0.66	0.07±0.25	0.00±0.00
h50N212a	0.00±0.00	1.90±1.91	2.60±2.55	0.05±0.22	2.77±1.92	0.07±0.25
r100N403a	0.00±0.00	0.00±0.00	0.00±0.00	0.60±0.60	0.13±0.35	0.03±0.18
r100N503a	0.00±0.00	0.50 ±0.71	0.00±0.00	0.85±0.67	0.17±0.38	6.57±16.08
r50N228a	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.10±0.31	0.03±0.18
r50N245a	0.05±0.22	0.30±0.48	0.00±0.00	0.35±0.49	0.80±0.89	0.23±0.50
w100N391a	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.03±0.18	7.23±6.18
w100N476a	0.00±0.00	0.00±0.00	0.00±0.00	0.45±0.69	0.60±0.81	4.73±11.69
w50N169a	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.03±0.18	0.00±0.00
w50N230a	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.27±0.49	1.37±4.54

was dominant through achievement of best results in two objectives. In contrast, algorithms used for comparison with FEPSO achieved best results mostly in only one objective. There are some exceptions where other algorithms achieved best results in two objectives, but achieved worst results in the third objective, which, to some extent, negatively affects their best achievement in two objectives. Such instances are h100N280a, where SimE gets best results for MU and NOC objectives, but shows worst results for the NUL objective. Another example is h50N148a where NSGA-II shows the same trends as that of SimE. There is only one test case of w50N169a where NSGA-II achieved the best results for two objectives (NOC and NUL) but did not achieve worst results for the MU objective. Note that in all results, there is only one instance where FEPSO showed the worst performance (NOC for

Table 13 Summary of test cases where different algorithms achieved best and worst results for the three objectives. Test cases where FEPSO achieved best results for two or all three objectives are highlighted in bold.

Test Case	MU		NOC		NUL	
	Best	Worst	Best	Worst	Best	Worst
h100N280a	SimE	SA	SimE	FEPSO	FEPSO, PDPSO, NSGA-II	SimE
h100N360a	FEPSO	SA	FEPSO	SA	FEPSO, WAPSO	SimE
h50N148a	SA	NSGA-II	SA	NSGA-II	FEPSO, PDPSO, WAPSO,SA	NSGA-II
h50N212a	SimE	WAPSO	FEPSO	NSGA-II	FEPSO	SimE
r100N403a	FEPSO, SimE	NSGA-II	FEPSO	NSGA-II	FEPSO, PDPSO, WAPSO	NSGA-II
r100N503a	FEPSO	SA	FEPSO	SA	FEPSO, WAPSO	SA
r50N228a	FEPSO, SimE	PDPSO	FEPSO	WAPSO	FEPSO, PDPSO, WAPSO, NSGA-II	SimE
r50N245a	FEPSO	PDPSO	FEPSO	SA	WAPSO	SimE
w100N391a	SimE	SA	FEPSO	SA	FEPSO, PDPSO, WAPSO, NSGA-II	SA
w100N476a	SimE	SA	FEPSO	SA	FEPSO, PDPSO, WAPSO	SA
w50N169a	SA	WAPSO	NSGA-II	PDPSO	FEPSO, PDPSO, WAPSO, NSGA-II, SA	SimE
w50N230a	FEPSO	PDPSO	FEPSO	SA	FEPSO, PDPSO, WAPSO, NSGA-II	SA

h100N280a). Based on the above discussion and observations, it can be fairly claimed that, overall, FEPSO showed the best performance compared to all other algorithms considered.

The overall better performance of FEPSO lies in its design, which combines the strong searching capabilities of PSO, augmented by the simulated evolution algorithm which allows a more intelligent local search capability. In contrast to this hybrid design of FEPSO, both SimE and SA lack efficient traversing of the whole search space, since both of them are local search algorithms after all. Furthermore, when compared with NSGA-II, PDPSO, and WAPSO, again the intelligent local search capability of FEPSO allowed it to outperform the three algorithms.

11 Conclusion

This paper proposed and investigated a multi-objective particle swarm optimization algorithm to efficiently solve the open shortest path first weight setting problem. Three optimization objectives, namely, maximum utilization, number of congested links, and number of unused links were considered in the optimization process. These conflicting objectives were aggregated into a scalar optimization function using the unified and-or fuzzy aggregation operator. The performance of the proposed algorithm was analyzed with regard to different algorithm parameters including swarm size, acceleration coefficients, inertia weight, and velocity clamping. Results revealed that swarm size, acceleration coefficients, and velocity clamping have a significant effect on the quality of results, but the algorithm was insensitive to variation in the inertia weight. Furthermore, a modified version of the fuzzy PSO, namely, fuzzy evolutionary PSO, was also proposed which incorporated characteristics of the simulated evolution algorithm. A comparison among the basic and modified versions of the PSO revealed that the fuzzy evolutionary PSO was able to produce results of higher quality compared to its basic counterpart. Furthermore, a comparison of fuzzy evolutionary PSO with Pareto-dominance PSO, weighted aggregation PSO, NSGA-II, simulated evolution, and simulated annealing algorithms revealed that the fuzzy evolutionary PSO outperformed the other five algorithms.

Appendix A

Nomenclature

G	Graph
N	Set of nodes

n	A single element in set N
A	Set of arcs
A^t	Set of arcs representing shortest paths from all sources to destination node t
a	A single element in set A. It can also be represented as (i, j)
s	Source node
v	Intermediate node
t	Destination node
D	Demand matrix
$D[s, t]$	An element in the demand matrix that specifies the demand from source node s to destination node t; It can also be specified as d_{st}
w_{ij}	Weight on arc (i, j); if $a = (i, j)$, then it can also be represented as w_a
c_{ij}	Capacity on arc (i, j); if $a = (i, j)$, then it can also be represented as c_a
Φ	Cost function
$\Phi_{i,j}$	Cost associated with arc (i, j); if $a = (i, j)$, then it can also be represented as Φ_a
δ_u^t	Outdegree of node u when destination node is t
$\delta^+(u)$	Outdegree of node u
$\delta^-(u)$	Indegree of node u
l_a^t	Load on arc a when destination node is t
l_a	Total traffic load on arc a
$f_a^{(s,t)}$	Traffic flow from node s to t over arc a
$SetCA$	Set of congested arcs

11.1 Terminology

1. A single element in the set N is called a “Node”. It is represented as n.

2. A single element in the set A is called an "Arc" or "Link". It is represented as a.
3. A set $G = (N, A)$ is a graph defined as a finite nonempty set N of nodes and a collection A of pairs of distinct nodes from N.
4. A "directed graph" or "digraph" $G = (N, A)$ is a finite nonempty set N of nodes and a collection A of ordered pairs of distinct nodes from N; each ordered pair of nodes in A is called a "directed arc".
5. A digraph is "strongly connected" if for each pair of nodes i and j there is a directed path $(i = n_1, n_2, ..., n_l = j)$ from i to j. A given graph G must be strongly connected for this problem.
6. A "demand matrix" is a matrix that specifies the traffic flow between s and t, for each pair $(s, t) \in N \times N$.
7. $(n_1, n_2, ..., n_l)$ is a "directed walk" in a digraph G if (n_i, n_{i+1}) is a directed arc in G for $1 \leq i \leq l - 1$.
8. A "directed path" is a directed walk with no repeated nodes.
9. Given any directed path $p = (i, j, k, ..., l, m)$, the "length" of p is defined as $w_{ij} + w_{jk} + ... + w_{lm}$.
10. The "outdegree" of a node u is a set of arcs leaving node u i.e., $\{(u, v) : (u, v) \in A\}$.
11. The "indegree" of a node u is a set of arcs entering node u i.e., $\{(v, u) : (v, u) \in A\}$.
12. The input to the problem will be a graph G, a demand matrix D, and capacities of each arc.
13. The term MU refers to the maximum utilization. It is the highest load/capacity ratio of the network.
14. The term NOC refers to the number of congested links.
15. The term NUL refers to the number of unused links.
16. The term E refers to the total number of links in the network.

Appendix B

Tables 14 to 25 provide the quality of solutions obtained with respect to the associated swarm size for all test cases. Column 1 represents the number of particles in the swarm. Column 2 represents the average overall goodness using the UAO operator. Column 3 represents the percentage difference between the average overall goodness of the corresponding number of particles and the highest average overall goodness (given in boldface) of the solutions. Note that the swarm size resulting in the highest average overall goodness is taken as the reference, and the difference for other swarm sizes is calculated with respect to the reference value. The differences were also statistically tested using Wilcoxon's rank sum test.

Table 14 Effect of swarm size on overall cost for *h100N280a* with UAO.

No. of particles	Fuzzy Cost (UAO)	% Difference
20	0.433±0.037	16.41*
40	0.471±0.018	9.07*
60	0.493±0.029	4.83*
80	0.504±0.019	2.7
100	**0.518**±0.025	NA

Table 15 Effect of swarm size on overall cost for *h100N360a* with UAO.

No. of particles	Fuzzy Cost (UAO)	% Difference
20	0.424±0.037	21.92*
40	0.48±0.034	11.6*
60	0.514±0.028	5.34*
80	0.529±0.027	2.58
100	**0.543**±0.036	NA

Table 16 Effect of swarm size on overall cost for *h50N148a* with UAO.

No. of particles	Fuzzy Cost (UAO)	% Difference
20	0.381±0.019	12.81*
40	0.4±0.019	8.47*
60	0.416±0.022	4.81*
80	0.422±0.019	3.43*
100	**0.437**±0.022	NA

Table 17 Effect of swarm size on overall cost for *h50N212a* with UAO.

No. of particles	Fuzzy Cost (UAO)	% Difference
20	0.399±0.06	20.83*
40	0.469±0.022	6.94*
60	0.483±0.026	4.17*
80	0.5±0.018	0.79
100	**0.504**±0.015	NA

Table 18 Effect of swarm size on overall cost for *r100N403a* with UAO.

No. of particles	Fuzzy Cost (UAO)	% Difference
20	0.385±0.019	19.96*
40	0.425±0.031	11.64*
60	0.448±0.022	6.86*
80	0.47±0.014	2.29*
100	**0.481**±0.017	NA

12 Compliance with Ethical Standards

The authors declare that they have no conflict of interest. This article does not contain any studies with human participants or animals performed by any of the authors. Informed consent was obtained from all individual participants included in the study.

Table 19 Effect of swarm size on overall cost for *r100N503a* with UAO.

No. of particles	Fuzzy Cost (UAO)	% Difference
20	0.41±0.014	24.49*
40	0.474±0.029	12.71*
60	0.506±0.022	6.81*
80	0.53±0.02	2.39
100	**0.543**±0.013	NA

Table 20 Effect of swarm size on overall cost for *r50N228a* with UAO.

No. of particles	Fuzzy Cost (UAO)	% Difference
20	0.444±0.018	18.23*
40	0.487±0.022	10.31*
60	0.512±0.023	5.71*
80	0.532±0.02	2.03
100	**0.543**±0.019	NA

Table 21 Effect of swarm size on overall cost for *r50N245a* with UAO.

No. of particles	Fuzzy Cost (UAO)	% Difference
20	0.444±0.025	19.86*
40	0.499±0.03	9.93*
60	0.54±0.021	2.53*
80	0.55±0.025	0.72
100	**0.554**±0.021	NA

References

1. K. G. Coffman and A. M. Odlyzko. Internet Growth: Is there a Moore's Law for Data Traffic? *Handbook of Massive Data Sets*, pages 47–93, 2001.
2. B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. *IEEE Conference on Computer Communications(INFOCOM)*, pages 519–528, 2000.

Table 22 Effect of swarm size on overall cost for *w100N391a* with UAO.

No. of particles	Fuzzy Cost (UAO)	% Difference
20	0.409±0.071	32.84*
40	0.49±0.063	19.54*
60	0.566±0.038	7.06*
80	0.582±0.044	4.43*
100	**0.609**±0.029	NA

Table 23 Effect of swarm size on overall cost for *w100N476a* with UAO.

No. of particles	Fuzzy Cost (UAO)	% Difference
20	0.489±0.025	25.57*
40	0.578±0.026	12.02*
60	0.618±0.032	5.94*
80	0.638±0.022	2.89*
100	**0.657**±0.025	NA

Table 24 Effect of swarm size on overall cost for *w50N169a* with UAO.

No. of particles	Fuzzy Cost (UAO)	% Difference
20	0.508±0.03	13.31*
40	0.541±0.019	7.68*
60	0.573±0.02	2.22
80	0.582±0.022	0.68
100	**0.586**±0.027	NA

3. J. F. Kurose and K.W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet*. Prentice Hall Series, 2002.

4. N. H. Bhagat. A New Hybrid Approach to OSPF Weight Setting Problem. *International Journal on Recent and Innovation Trends in Computing and Communication*, 1(5):443–450, 2013.

Table 25 Effect of swarm size on overall cost for *w50N230a* with UAO.

No. of particles	Fuzzy Cost (UAO)	% Difference
20	0.4±0.084	31.51*
40	0.504±0.038	13.7*
60	0.538±0.044	7.88*
80	0.564±0.036	3.42
100	**0.584**±0.043	NA

5. F.Bizri and B. Sanso. Corouting: an IP Hybrid Routing Approach. In *Fourth International Conference on Networking and Services*, pages 46–52, 2008.
6. S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the Tightrope: Responsive Yet Stable Traffic Engineering. In *ACM 2005 conference on applications, technologies, architectures, and protocols for computer communications*, pages 253–264, 2005.
7. E. W. Dijkstra. A Node on Two Problems in Connection of Graphs. *Numerical Mathematics*, 1959.
8. B. Fortz and M. Thorup. Increasing Internet Capacity using Local Search. *Technical Report IS-MG*, 2000.
9. R. Kling and P. Banerjee. Optimization by Simulated Evolution with Applications to Standard Cell Placement. *In Proceedings of 27th Design Automation Conference*, pages 20–25, 1990.
10. B. Fortz and M. Thorup. Optimizing OSPF/IS-IS Weights in a Changing World. *IEEE Journal on Selected Area in Communications*, 20(4):756–767, September 2006.
11. M. H. Sqalli, S. M. Sait, and M. A. Mohiuddin. An Enhanced Estimator to Multi Objective OSPF Weight Setting Problem. *Network Operations and Management Symposium, NOMS*, 2006.
12. M. Rodrigues and K. G. Ramakrishnan. Optimal Routing in Data Networks. *Bell Labs Technical Journal*, 6(1):117–138, 2002.
13. M. Ericsson, M. G. C. Resende, and P. M. Pardalos. A Genetic Algorithm for the Weight Setting Problem in OSPF Routing. *J. Combinatorial Optimisation conference*, 2002.
14. M. Zagozdzon, M. Dzida, and M. Pioro. Traffic Flow Optimization in Networks with Combined OSPF/MPLS Routing. In *IEEE 15th International Conference on Advanced Computing and Communications*, pages 131–137, 2007.

15. A. Abo Ghazala and A. El Sayed. Open Shortest Path First Weight Setting (OSPFWS) solving algorithms comparison and new method for updating weights. *International Journal of Computer Science and Network Security*, 9(5):191–197, May 2009.

16. A. Parmar, S. Ahmed, and J. Sokol. *An integer programming approach to the OSPF weight setting problem*. NSF Technical Report no. DMI-0457066, 2006.

17. S. Srivastava, G. Agarwal, D. Medhi, and M. Pioro. Determining feasible link weight systems under various objectives for OSPF networks. *IEEE eTransactions on Network and Service Management*, 2(1), 2005.

18. L. Buriol, M. Resende, C. Rebeiro, and M. Thorup. TA memetic algorithm for OSPF routing. In *6th INFORMS Telecom*, pages 187–188, 2002.

19. A. Bley. On the approximability of the minimum congestion unsplittable shortest path routing problem. In *Integer Programming and Combinatorial Optimization (IPCO 2005), Lecture Notes in Computer Science LNCS*, pages 97–210, 2005.

20. A. Bley. Approximability of unsplittable shortest path routing problems. *Networks*, 54(1):23–46, 2009.

21. L. Lin and M. Gen. Priority-Based Genetic Algorithm for Shortest Path Routing Problem in OSPF. *Intelligent and Evolutionary Systems, Studies in Computational Intelligence*, 187:91–104, 2009.

22. M. Pioro, A. Szentsi, J. Harmatos, A. Juttner, P. Gajownicczek, and S. Kozdrowski. On open shortest path first related network optimization problems. *Performance Evaluation*, 48(4):201–223, 2002.

23. G. Retvari and T. Cinkler. Practical ospf traffic engineering. *IEEE Commununications Letters*, 8:689–691, 2004.

24. G. Retvari, J. Biro, and T. Cinkler. On Improving the Accuracy of OSPF Traffic Engineering. In *NETWORKING 2006, Lecture Notes in Computer Science (LNCS), Vol. 3976*, pages 51–62, 2006.

25. A. Nucci, S. Bhattacharyya, N. Taft, and C. Diot. Igp link weight assignment for operational tier-1 backbones. *IEEE/ACM Transactions on Networking*, 15(4):789–802, 2007.

26. G. Shrimali, A. Akella, and A. Mutapcic. Cooperative interdomain traffic engineering using nash bargaining and decomposition. *IEEE/ACM Transactions on Networking*, 18(2):341–352, 2010.

27. A. Riedl. Optimized routing adaptation in IP networks utilizing OSPF and MPLS. In *IEEE International Conference on Communications*, pages 1754–1758, 2003.

28. S. Lee, T. Po-Kai, and A. Chen. Link weight assignment and loop-free routing table update for link state routing protocols in energy-aware internet. *Future Generation Computer Systems*, 28:437–445, 2012.

29. B. Fortz, J. Rexford, and M. Thorup. Traffic Engineering with Tradional IP Routing Protocols. *IEEE Communications Magazine*, pages 118–124, 2002.

30. F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

31. D. Frigioni, M. loffreda, U. Nanni, and G. Pasqualone. Experimental Analysis of Dynamic Algorithms for the Single Source Shortest Paths Problem. *ACM Journal of Experimental Algorithms*, 1998.

32. G. Ramalingam and T. Reps. An Incremental Algorithm for a Generalization of the Shortest Path Problem. *Journal of Algorithms*, pages 267–305, 1996.

33. B. Fortz. Combinatorial Optimization and Telecommunications. *http://www.poms.ucl.ac.be/staff/bf/en/COCom-5.pdf*.

34. J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

35. A. Abo Ghazala, A. El Sayed, and M. Mousa. A Survey for Open Shortest Path First Weight Setting (OSPFWS) Problem. *The 2nd International Conference on Information Security and Assurance (ISA2008)*, pages 24–26, April 2008.

36. R. Reis, M. Ritt, L. Buriol, and M. Resende. A memetic algorithm for the weight setting problem in DEFT. In *COMCEV 2007*, pages 1–6, 2007.

37. A. Abo Ghazala, A. El Sayed, and M. Mousa. A New Approach for Open Shortest Path Weight Setting (OSPFWS) Problem. *Convergence and Hybrid Information Technology*, pages 188 – 193, November 2008.

38. S. M. Sait, M. H. Sqalli, and M. A. Mohiuddin. Engineering Evolutionary Algorithm to Solve Multi Objective OSPF Weight Setting Problem. *Australian Conference on Artificial Intelligence*, pages 950–955, 2006.

39. P. Laarhoven and E. Aarts. *Simulated Annealing: Theory and Applications*. Kluwer Academic, Norwell, Massachusetts, 1987.

40. R. Kling and P. Banerjee. Empirical and Theoretical Studies of the Simulated Evolution Method Applied to Standard Cell Placement. *IEEE Transactions on Computer-Aided Design*, 10(10):1303–1315, October 1991.

41. M. Houssaini Sqalli, S. Mohammed Sait, and S. Asadullah. Minimizing the Number of Congested Links in OSPF Routing. *ATNAC*, December 2008.

42. M. Mohiuddin, S. A. Khan, and A. P. Engelbrecht. Simulated evolution and simulated annealing algorithms for solving multi-objective open shortest path first weight setting problem. *Applied Intelligence, Springer*, 40(3):1–20, 2014.

43. L. A. Zadeh. Fuzzy Sets. *Information Control*, 8:338–353, 1965.

44. L. A. Zadeh. The Concept of a Linguistic Variable and its Application to Approximate Reasoning. *Information Sciences*, 8:199–249, 1975.

45. L. A. Zadeh. Outline of a New Approach to the Analysis of Complex Systems and Decision Processes. *IEEE Transaction Systems, Man, and Cybernetics*, 3(1):28–44, 1973.

46. H. Li and V. Yen. *Fuzzy Sets and Fuzzy Decision-Making*. CRC Press, USA, 1995.

47. H. Hamacher. Ueber Logische Verknupfungen Unschalfer Aussagen und deren Zugehoerige Bewertungs-funktione. *Progress in Cybernetics and Systems Research*, 3:276–288, 1978.

48. M. Frank. On the Simultaneous Associativity of $F(x, y)$ and $x + y - F(x, y)$. *Aequationes Mathematicae*, 19:194–226, 1979.

49. S. Weber. A General Concept of Fuzzy Connectives, Negations and Implications Based on t-Norms and t-Conorms. *Fuzzy Sets & Systems*, 11:115–134, 1983.

50. D. Dubois and H. Prade. Operations in Fuzzy-valued Logic. *Information and Control*, 43:224–240, 1979.

51. S. A. Khan and A. P. Engelbrecht. A New Fuzzy Operator and its Application to Topology Design of Distributed Local Area Networks. *Information Sciences*, 177(12):2692–2711, 2007.

52. J. Kennedy and R. C. Eberhart. Particle Swarm Optimization. *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

53. R. Kling and P. Banerjee. Empirical and Theoretical Studies of the Simulated Evolution Method Applied to Standard Cell Placement. *IEEE Transactions on Computer-Aided Design*, pages 1303–1305, October 1991.

54. R. Kling and P. Banerjee. ESP: Placement by Simulated Evolution. *IEEE Transactions on Computer-Aided Design*, pages 245–255, March 1989.

55. E. W. Zegura. GT-ITM: Georgia Tech Internetwork Topology Models (software). *http://www.cc.gatech.edu/faq/Ellen.Zegura/gt-itm/gt-itm.tar.gz*, 1996.

56. K. Calvert, M. Doar, and E. W. Zegura. Modeling Internet Toplogy. *IEEE Communications Magazine*, (35):160–163, 1997.

57. E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How To Model An Internetwork. *15th IEE Conference on Computer Communications (INFOCOM)*, pages 594–602, 1996.

58. F. van den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD Thesis, University of Pretoria, 2001.

59. S. A. Khan and A. P. Engelbrecht. A fuzzy particle swarm optimization algorithm for computer communication network topology design. *Applied Intelligence*, 36:161–177, 2012.

60. H. Cho, B. Wang, and S. Roychowdhury. Automatic Rule Generation for Fuzzy Controllers using Genetic Algorithms: A Study on Representation Scheme and Mutation Rate. In *IEEE World Congress on Computational Intelligence*, pages 1290–1295, 1998.

61. R. Haupt. Optimum Population Size and Mutation Rate for a Simple Real Genetic Algorithm that Optimizes Array Factors. In *IEEE Antennas and Propagation Society International Symposium*, pages 1034 – 1037, 2000.

62. M. Lim, S. Rahardja, and B. Gwee. A GA Paradigm for Learning Fuzzy Rules. *Fuzzy Sets & Systems*, 82:177–186, 1996.

63. J. Liska and S. S. Melsheimer. Complete Design of Fuzzy Login System using Genetic Algorithms. In *3rd IEEE International Conference on Fuzzy Systems*, pages 1377–1382, 1994.

64. J. Alvarez-Benitez and R. Everson and J. Fieldsend. A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts. In *3rd International Conference on Evolutionary Multi-criterion Optimization, Lecture Notes in Computer Science (LNCS)*, 3410:459–473, 2005.

65. K. Parsopoulos and M. Vrahatis. Particle swarm optimization method in multiobjective problems. In *ACM Symposium on Applied Computing*, pages 603–607, 2002.

66. K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.