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

Simulated Evolution and Simulated Annealing Algorithms for Solving Multi-objective Open Shortest Path First Weight Setting Problem

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

Received: date / Accepted: date

Abstract Optimal utilization of resources in present-day communication networks is a challenging task. Routing plays an important role in achieving optimal resource utilization. The open shortest path first (OSPF) routing protocol is widely used for routing packets from a source node to a destination node. This protocol assigns weights (or costs) to the links of a network. These weights are used to determine the shortest path between all sources to all destination nodes. Assignment of these weights to the links is classified as an NP-hard problem. This paper formulates the OSPF weight setting problem as a multi-objective optimization problem, with maximum utilization, number of congested links, and number of unused links as the optimization objectives. Since the objectives are conflicting in nature, an efficient approach is needed to balance the trade-off between these objectives. Fuzzy logic has been shown to efficiently solve multi-objective optimization problems. A fuzzy cost function for the OSPF weight setting problem is developed in this paper based on the Unified And-OR (UAO) operator. Two iterative heuristics, namely, simulated annealing (SA) and simulated evolution (SimE) have been implemented to solve the multi-objective OSPF weight setting problem using a fuzzy cost function. Results are compared with that found using other cost functions proposed in the liter-

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

Salman A. Khan
Department of Computer Engineering, College of Information Technology, University of Bahrain, Sakhir, Bahrain
Department of Computer Science, University of Pretoria, Pretoria 0002, South Africa E-mail: sakhan@uob.edu.bh

Andries P. Engelbrecht
Department of Computer Science, University of Pretoria, Pretoria 0002, South Africa
E-mail: engel@cs.up.ac.za

ature [1]. Results suggest that, overall, the fuzzy cost function performs better than existing cost functions, with respect to both SA and SimE. Furthermore, SimE shows superior performance compared to SA. In addition, a comparison of SimE with NSGA-II shows that, overall, SimE demonstrates slightly better performance in terms of quality of solutions.

Keywords Open Shortest Path First Algorithm; Optimization; Fuzzy logic; Simulated evolution; Simulated Annealing; Routing.

1 Introduction

Internet traffic is increasing rapidly due to the increase in web based applications [2]. To deal with this high volume of traffic, efficient utilization of network resources, such as network bandwidth, is essential. The primary objective of network traffic engineering is to map traffic efficiently on the available network resources, preventing traffic imbalance if it exists [3].

Routers are the main interconnection points of the internet. They forward data packets between source and destination nodes through multiple paths which exist between a given source and destination pair. The complexity of the internet is due to its huge size. The internet is divided into autonomous systems (AS) to manage its complexity. An AS is a collection of networks under the control of one single entity or organization with a specific routing policy. These policies are defined by a class of routing protocols, namely, interior gateway protocols (IGP) [4]. Routing across ASs is performed by another class of protocols, namely, exterior gateway protocols (EGP) [4]. Open shortest path first (OSPF) [4] is an IGP. The foundation of OSPF is based on Dijkstra's algorithm [5], which determines a shortest path between a source and destination pair. The metric for shortest path determination is the link weight. The cost of path between a given source and destination pair is the sum of OSPF weights on the links in that path. The path with least cost is considered as the shortest path.

This paper focuses on the open shortest path first weight setting (OSPFWS) problem, which is an NP-hard problem [6]. The OSPFWS problem requires that a set of weights be determined such that network resources are utilized efficiently. The objectives of this problem is to minimize maximum utilization, minimize number of congested links, and minimize number of unused links. These objectives are conflicting in nature, i.e. if improvement in one objective is obtained, at least one of the other objectives may result in deterioration. To address this NP-hard problem with conflicting objectives, this paper proposes to apply two algorithms, namely, a fuzzy logic based simulated annealing [7] and a fuzzy logic based simulated evolution [8] algorithm. The performance of these two algorithms is compared. Results are also compared with other formulations of the objective function.

The rest of the paper is organized as follows: Section 2 provides the necessary background and related work on the OSPFWS problem, as well as an introduction to the simulated annealing and simulated evolution algorithms. Section 3 gives the formal definition of the OSPFWS problem. Section 4 provides a primer on fuzzy logic and the Unified And-OR operator. The formulation of a fuzzy logic based objective function for the OSPFWS problem is covered in Section 5. Section 6 describes the proposed algorithms. Results are provided and discussed in Section 7. The paper is concluded in Section 8. Finally, symbols and terminology used in this paper are given in the appendix.

2 Background and Related Work

Literature has reported considerable research on optimizing OSPF weights [1,3,9,10]. The term “Maximum utilization” is defined as the maximum value of all the utilization values over all the links in the network. Fortz and Thorup [3] were the first to formulate a cost function based on utilization ranges, and applied tabu search [11] to optimize (i.e. minimize) “maximum utilization”.

The formal definition of the Fortz and Thorup cost function is as follows:

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

subject to the constraints:

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

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

The constraint in Equation (2) implies that total traffic load on arc a is equal to the sum of the traffic load on arc a and the traffic load on all the incoming arcs to arc a. Equation (3) signifies 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 are piecewise linear functions, with $\Phi_a(0) = 0$ and a derivative, $\Phi_a^{'}(l_a)$ given by (see Figure 1):

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

(4)

This function signifies that the utilization (i.e. load to capacity ratio) of a link is acceptable within 100% of the link's capacity. The cost assigned to the links having a utilization level within 100% of the link capacity is 1, 3, 10, or 70, depending on the level of utilization. For example, if utilization is equal to or more than 0% but less than 33.33%, then a cost of 1 is assigned; for utilization of over 33.33% but up to 66.66%, a cost of 3 is assigned, and so on). On the other hand, if the utilization of a link is beyond 100% (which means that more packets are coming to the link beyond its maximum capacity) then such an over-utilization is not acceptable, since it will result in packets being dropped. 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%.

The approach followed by Fortz and Thorup was based on the observation that equal traffic load distribution across the links can be achieved by multiple equidistance shortest paths between the source and the destination node. Fortz and Thorup obtained multiple shortest paths using a dynamic shortest path algorithm [12–14]. Ramakrishnan and Rodrigues [9] proposed a local search procedure similar to that of Fortz and Thorup. Although Ramakrishnan and Rodrigues used the same cost function as Fortz and Thorup, the main difference between the two approaches was that Rodrigues and Ramakrishnan's technique increases the link metric (the cost or OSPF weight assigned to a link) of a heavily used link. This was done as an attempt to reach neighbor solutions in the search space. In addition, Rodrigues and Ramakrishnan used a different test case to that used by Fortz and Thorup. Ericsson *et al.* [10] applied a genetic algorithm [15] to the OSPFWS problem using the cost function proposed by Fortz and Thorup.

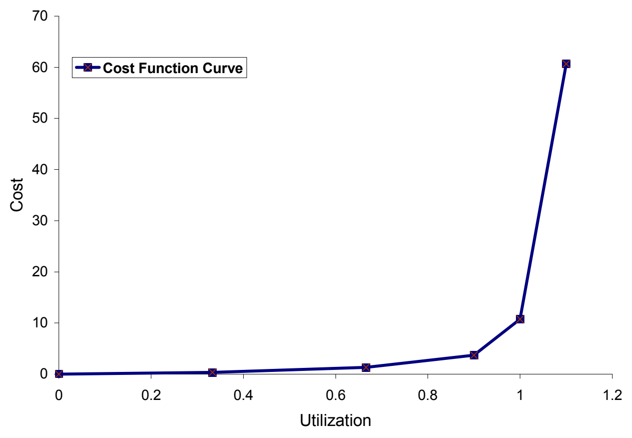


Fig. 1 Cost function curve [3].

Sqalli *et al.* [1] proposed minimization of the number of congested links using SA [16] in order to solve the OSPFWS. In this work, this cost function is denoted as SqalliCF and defined as

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

where MU is the maximum utilization of any link in the network, which is the maximum value obtained from the load to capacity ratio of all the links. $SetCA$ refers to the set of congested links, l_a is the total traffic on link a, c_a is the capacity of link a, and E represents the total number of links in the network. The second term of Equation (5) after the plus sign represents the extra load on the network. This extra load is taken from all the congested links, divided by the total number of links present in the network to normalize the entire function. If there is no congestion in the network, the whole term after the plus sign will be zero. Thus, Equation (5) results in minimization of maximum utilization if there is no congestion in the network. If congestion exists, then the function results in minimization of maximum utilization as well as 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 [3]. Sqalli *et al.* also found the maximum utilization to be comparable to that obtained by the approach of Fortz and Thorup. Using the cost function of Equation (5), Sqalli *et al.* applied the SimE algorithm [17] to the OSPFWS problem and compared the results with the results of SA [18]. Tabu search using SqalliCF [1] has also been applied to the OSPFWS problem [19].

Abo Ghazala *et al.* [20] have surveyed various algorithm applications to the OSPFWS problem. They have also applied a different technique to the OSPFWS problem [21]. This technique increases the weight values for links with higher utilization, decreases weight values for links with low utilization, and increases weight values for links with less empty bandwidth (difference between link capacity and traffic on the link). Abo Ghazala *et al.* used the same cost function as that of Fortz and Thorup (refer to Equation (1)), but employed iterative local search. The test cases were small and also different from the test cases used in this paper. Results were compared with the Cisco weight setting criteria [21], unity weight setting, and random weight setting.

The cost function proposed by Fortz and Thorup (Equation (1)), which was subsequently used by many researchers [1,9,10,18,21] as discussed above, is based on utilization ranges. The aim of the cost function of Fortz and Thorup was to minimize "maximum utilization" only. Minimizing "maximum utilization" only might lead to the existence of congested and unused links. The cost function proposed by Sqalli (Equation (5)) was aimed at optimizing maximum utilization and the number of congested links. However, optimizing these two factors does not necessarily guarantee that the number of unused links will definitely be optimized implicitly, because SqalliCF does not say anything about optimizing NUL. Therefore, it would be more logical to also use unused links in the optimization process. This can be achieved by shifting the traffic of congested links to unused links. Therefore, the cost function proposed in this paper is motivated by the above observations, and therefore addresses the simultaneous optimization of maximum utilization, number of congested links, and number of unused links through the use of fuzzy logic, as mentioned earlier.

2.1 Simulated Annealing Algorithm

Simulated annealing (SA) is a popular heuristic algorithm proposed by Kirkpatrick *et al.* [7]. It is derived from the analogy of the physical annealing process of metals. SA is applied to a single solution, where each perturbation of the current solution results in a single move. The core of the algorithm is the metropolis procedure, which perturbs the existing solution by making a move, thus resulting in a new solution. Solutions which result in a reduced cost are readily accepted, while solutions with higher cost than the previous solution is accepted probabilistically based on the *metropolis criterion.* Several parameters such as initial temperature T_0, cooling rate α, and markov chain length M play a key role in the performance and convergence of the algorithm, and therefore need a very careful tuning in order to produce solutions of highest quality. The SA algorithm and the metropolis procedure are illustrated in Figures 2 and 3 respectively. Interested reader may refer to [22] for further details.

2.2 Simulated Evolution Algorithm

Simulated evolution (SimE) is a general search strategy (since it can be applied to any complex optimization problem irrespective of the domain) proposed by Kling and Banerjee [8, 23, 24]. SimE is derived from the analogy of the biological evolution process which results in heritable changes in a population spread over many generations. The pseudo-code of SimE is given in Figure 4. Like SA, SimE also operates on a single solution, where a solution is comprised of elements (also referred to as individuals). The algorithm iterates between the *evaluation, selection,* and *allocation* phases. In the evaluation phase, goodness (i.e. fitness) of each element of a solution is evaluated based on a goodness function. Based on the outcome of the evaluation phase, elements are selected in the selection phase so that they can be replaced with newer elements. The selection is done probabilistically based on a selection function and through a parameter called the *Bias*, denoted by B. The replacement of older elements with newer ones is done

```
**ALGORITHM** Simulated_Annealing($S_0, T_0, \alpha, \beta, M, Maxtime$);
        (*$S_0$ is the initial solution *)
        (*Sbest is the best solution *)
        (*$T_0$ is the initial temperature *)
        (*$\alpha$ is the cooling rate *)
        (*$\beta$ is a constant *)
        (*$Maxtime$ is the total allowed time for the annealing process *)
        (*$M$ represents the time until the next parameter update *)
Begin
        $T = T_0$;
        $CurS = S_0$;
        $Sbest = CurS$;  /* Sbest is the best solution seen so far */
        $Ccur = \text{Cost}(CurS)$;
        $Cbest = \text{Cost}(Sbest)$;
        $Time = 0$;
                Repeat
                        Call Metropolis($CurS$, $Ccur$, $Sbest$, $Cbest$, $T$, $M$);
                        $Time = Time + M$;
                        $T = \alpha T$;
                        $M = \beta M$
                Until ($Time \geq MaxTime$);
                Return($Sbest$)
End (*of Simulated_Annealing*)
```

Fig. 2 Procedure for the simulated annealing algorithm (adopted from [22]).

in the allocation phase, in which a compound move is performed, thus replacing all selected elements in a single iteration. A detailed discussion on the simulated evolution algorithm can be found in [22].

The motivation behind using SA and SimE to solve the OSPFWS problem is threefold. Firstly, SA and SimE have shown relatively better performance than the local search approach of Fortz and Thorup in terms of optimizing two objectives, i.e. minimizing “maximum utilization” and “number of congested links” [1]. Secondly, SA and SimE have already been applied to a simpler version of the OSPFWS problem than the one proposed in this paper, and it would be of interest to build upon and improve already existing approaches.

```
**ALGORITHM** Metropolis($CurS$, $Ccur$, $Sbest$, $Cbest$, $T$, $M$);
 **Begin**
     **Repeat**
         $NewS = Neighbor(CurS)$;
         $Cnew$ = Cost($NewS$);
         $\Delta\ Cost = Cnew$ - $Ccur$;
         **If** ($\Delta\ Cost < 0$) **Then**
             $CurS = NewS$;
                 **If** $Cnew < Cbest$ **Then**
                     $Sbest = NewS$
                 **Endif**
         **Else if** ($RANDOM < e^{-\Delta Cost/T}$) **Then**
                 $CurS = New$;
         **Endif**
         $M = M - 1$
     **Until** ($M = 0$)
**End** (*of Metropolis*)
```

Fig. 3 The Metropolis procedure (adopted from [22]).

Note that, for the approaches proposed in this paper, a third objective (i.e. number of unused links) is also included in the optimization process, in addition to the two objectives mentioned above. Moreover, the conflicting nature of these three objectives have been addressed by employing fuzzy logic. Thirdly, both SimE and SA have been applied to solve various multi-objective optimization problems. Some examples for SA are [25–28], and for SimE are [29–31]. Thus, the overall aim of this paper is to compare and study the performance of fuzzy SA and fuzzy SimE algorithms (with three optimization objectives) with respect to the existing SA and SimE approaches (with two objectives).

3 OSPFWS 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 on calculation of traffic load on links.

```
**ALGORITHM** *Simulated_Evolution(M, L)*;
/* $M$: Set of movable elements; */
/* $L$: Set of locations; */
/* $B$: Selection bias; */
/* Stopping criteria and selection bias can be automatically adjusted; */
$INITIALIZATION$;
**Repeat**
    $EVALUATION$:
        **Foreach** $m \in M$ **Do**
            $g_m = \frac{O_m}{C_m}$
        **End Foreach**;

    $SELECTION$:
        **Foreach** $m \in M$ **Do**
            **If** *Selection(m,B)* **Then** $P_s = P_s \cup \{m\}$
                **Else** $P_r = P_r \cup \{m\}$
            **Endif**;
        **End Foreach**;
    *Sort the elements of* $P_s$;

    $ALLOCATION$:
        **Foreach** $m \in P_s$ **Do**
            *Allocation*$(m)$
        **End Foreach**;
**Until** *Stopping-criteria are met*;
**Return**($BestSolution$);
**End** *Simulated_Evolution*.
```

Fig. 4 The simulated evolution algorithm (adopted from [22]).

```
**Function** *Selection(m,B)*;
/* $m$: is a particular movable element; */
/* $B$: Selection Bias; */
        **If** $Random \leq 1 - g_m + B$ **Then Return** True
            **Else Return** False
        **Endif**
**End** *Selection*;
```

Fig. 5 Selection function employed in the SimE algorithm of Figure 4 (adopted from ([22]).

3.1 OSPF Weight Setting Problem

The OSPF weight setting (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 $w_a \in [1, w_{max}]$ for each arc $a \in A$ such that the objective function or cost function Φ is minimized. w_{max} is a user-defined upper limit. Fortz and Thorup [32] found that a small set of weight values significantly reduces the overhead of the algorithm. By experimentation, w_{max} was set to 20. The chosen arc weights 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 depends highly on the choice of weights. Figure 6 depicts a topology with weights assigned within the range $[1, 20]$. A solution for this topology can be $(18, 1, 7, 15, 3, 17, 14, 19, 13, 18, 4, 16, 16)$. These elements (i.e. weights) are arranged in a specific order for simplicity: The outgoing links from node A are listed first (i.e. AB, AF), followed by the outgoing links from node B (i.e. BC, BD), and so on (in other words, a breadth-first expansion).

For the purposes of this paper, three objectives have to be simultaneously optimized. These objectives are maximum utilization, number of congested links, and number of unused links, all of which need to be minimized. Minimizing maximum utilization may lead to better distribution of network traffic across all the links such that congestion can be avoided and network resources can be utilized well [6]. Network administrators prefer to keep links less congested. However, if there is congestion in the network, then it is preferred to reduce the congestion by at least minimizing the total number of congested links. Suppose there is a network having 40 congested links, and also having 15 unused links. It would be preferred to accommodate the traffic of the 40 congested links on the existing 15 unused links. Thus, minimizing the number of unused links also affects the performance of the network. This is because traffic distribution across the links of the networks depends on the routing paths established. A new solution might therefore create new routing paths such that traffic on congested links may be distributed on unused links.

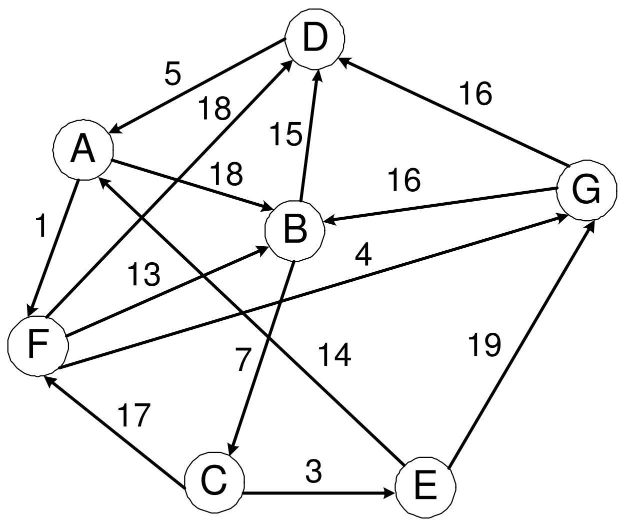


Fig. 6 Representation of a topology with assigned weights.

3.2 Traffic Load Calculation

This section provides details of the steps of calculating 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 [5]. 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 algo-

rithm 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

Fuzzy logic [33,34] has been extensively applied to a huge number of multi-objective optimization and decision-making problems in a variety of domains. The original 'And' function proposed by Zadeh used the "min" function. This implementation of 'And' was a very rigid approach since it only considered the worst membership value in the optimization process, while completely neglecting the positive effects of better membership values of other objectives. This observation subsequently led to the development of several 'soft-And' operators that would consider the effects of all design objectives in the decision-making process by aggregating all membership values into a single optimization function. Some well-known aggregation operators are the ordered weighted average operator [35], Einstein's operator [36], Hamacher's operator [37], Frank's operator [38], Weber's operator [39], Dubois and Prade's operator [40], and the unified AND-OR operator [29], among others. These operators allow easy adjustment of the degree of "anding" embedded in the aggregation.

The Unified And-Or (UAO) operator, proposed by Khan and Engelbrecht [29], has an important characteristic which is that a single equation is used to adjust the degree of "anding" and "oring" embedded in the aggregation, 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 using the UAO operator is controlled by a variable $\nu \geq 0$, whose value determines whether the function behaves as AND or OR. The operator is defined as:

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

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 detail on the UAO operator, the interested reader is referred to Khan and Engelbrecht [29].

5 Fuzzy Logic approach to the OSPFWS Problem

The OSPFWS problem can be solved by assigning a set of weights to the network links. The best solution to the problem is one which optimizes network resources efficiently. The objectives of OSPFWS problem include maximum utilization (MU), the number of congested links (NOC) and the number of unused links (NUL). These objectives, if considered individually and separately, do not provide adequate information for deciding the quality of a solution, since they are interdependent. Therefore, it is more appropriate to consider their combined effect to assess the quality of a solution. The conflicting nature of these objectives further amplifies the complexity of the problem. Under such a situation, a mechanism is required to find a set of solutions (referred to as Pareto solutions) that balance the trade-off among the objectives. Fuzzy logic provides a mechanism to conveniently deal with the trade-off among multiple, conflicting objectives.

The rest of this section details the use of fuzzy logic to combine the three conflicting objectives into a single overall objective. This single 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, by using membership functions. This needs the formulation of membership functions for each individual objective. This process is detailed below.

To define the membership function of maximum utilization, two extreme values, the upper and lower bounds, are determined first. Figure 7 shows the membership function of the objective to be optimized. If the objective is maximum utilization, then point ‘A’ refers to the minimum MU (MinMU) and point ‘B’ refers to the maximum MU (MaxMU). The membership value for MU, μ_{MU}, is determined as follows:

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

The membership function for NOC, μ_{NOC}, is defined in a similar way. In Figure 7, point 'A' then refers to the minimum NOC (MinNOC) and 'B' refers to the maximum NOC (MaxNOC). The membership function of NOC is defined as follows:

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

Finally, the membership function for NUL, μ_{NUL}, is also represented by Figure 7, where the minimum (MinNUL) and maximum (MaxNUL) values correspond to 'A' and 'B', respectively. The membership value for NUL is determined as follows:

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

Since the aim is to obtain a solution having minimum values of MU, NOC, and NUL, a fuzzy rule can be formulated as follows:

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

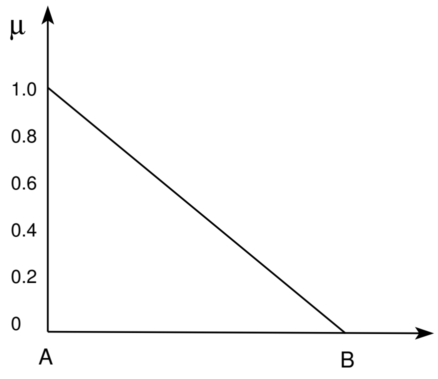


Fig. 7 Membership function of the objective to be optimized

The words 'low MU', 'low NOC', 'low NUL', and 'good solution' are linguistic values, each defining a fuzzy subset of solutions. Using the UAO operator [29], 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 $i = \{1, 2, 3\}$ represent 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 $\mu_{MU} = 0.13$, $\mu_{NOC} = 0.4$, and $\mu_{NUL} = 0.1$. Also assume that $\nu = 0.5$. Then, Equation (10) will result in a value of 0.228. Similarly, consider $\mu_{MU} = 0.13$, $\mu_{NOC} = 0.5$, and $\mu_{NUL} = 0.4$ associated with another arbitrary solution S_2. Again assume that $\nu = 0.5$. Then, the value of Equation (10) will be 0.276. Thus, solution S_2 is a better solution than solution S_1 in terms of quality. Equation (10) is employed as a fuzzy cost function for solving the OSPFWS problem using the SA and the SimE algorithms. This paper denotes the proposed fuzzy cost function as FuzzyCF.

6 Proposed Algorithms and Implementation Details

This section describes the details of the proposed fuzzy SA and fuzzy SimE algorithms to solve the OSPFWS problem.

6.1 Proposed Fuzzy Simulated Annealing Algorithm

As mentioned earlier, the OSPFWS problem requires assignment of a set of weights to the links in the network. Thus, the weights are the movable elements. A change in the weight of a link leads to different shortest paths between source and destination pairs, resulting in a new solution, and thus providing a different cost. Below, each step of the proposed fuzzy SA algorithm for the OSPFWS is described.

Initialization

The initialization phase generates a random solution (i.e. a set of weights). The SA control parameters, i.e. the initial temperature T_0, the cooling rate α, the constant β, maximum time for the annealing process $MaxTime$ (in terms of number of iterations), and the length of markov chain M are also initialized.

The SA control parameters have an impact on the convergence of the algorithm. For example, the initial temperature T_0 is set to an appropriate value, so that all moves are accepted initially. A very high initial temperature causes the algorithm to search blindly. On the other hand, a very low initial temperature results in bad solutions to be rejected in the early stage of the search. A number of methods have been proposed in the literature to determine an appropriate initial temperature [7,41–44]. This paper adopts the method proposed by Kirkpatrick [7], where the value of T_0 is chosen such that the initial acceptance ratio $X(T_0)$ is close to unity. The acceptance ratio is calculated using

$$X(T_0) = \frac{Number\ of\ moves\ accepted\ at\ T_0}{Total\ number\ of\ moves\ attempted\ at\ T_0} \quad (11)$$

The cooling rate, α, controls the rate of decrease in temperature. The higher the value of α, the lower the decrease in temperature, and vice versa. A typical value of α ranges from 0.8 to 0.99 [16].

Parameter M has also an impact on the convergence of SA, and it is very important to determine an appropriate value of M. A very high value of M increases the execution time. As an example, consider $M = 35$. This means that, at a given temperature, the algorithm will attempt 35 moves. It is quite probable that the same quality of solution is achieved with $M = 15$. Therefore, 20 moves are wasted. Similarly, when M is very low (for example, 5) then the algorithm may also result in bad quality solutions, simply because of the fact that the algorithm did not have sufficient time to explore the search space.

Metropolis Procedure

The core procedure of the annealing algorithm is the metropolis procedure. This procedure is repeated for a given number of iterations. During the metropolis procedure, a solution is perturbed. For the OSPFWS problem, a move involves selecting a weight from the set of given weights, and replacing the selected weight with a different weight value. For example, with reference to Figure 6, replacement of weight value 17 of the link between node A and node E by a new weight value of 10 is a single move. This replacement results in a new solution. The cost of this solution is evaluated and compared with the previous solution. If the new cost is better than the old cost, then the new solution is accepted. If the cost of the new solution is less than the cost of the previous solution, then the new solution is probabilistically accepted as explained in Section 2.

Evaluation of a solution

Once a new solution is generated, its cost should be computed. This is done using FuzzyCF (Equation (10)). For each individual objective, minimum and maximum values are a prerequisite for obtaining the membership values. These extreme values are found as follows: For the MU objective, the minimum value, 'MinMU', is taken as the minimum utilization of the initial solution given to the algorithm during the initialization phase. The maximum value for MU, 'MaxMU', is taken as the maximum utilization of the initial solution. The minimum value for the NOC objective, 'MinNOC', is set to zero. The maximum value for the NOC objective, 'MaxNOC', is taken as the number of congested links obtained from the initial solution. In the same way, 'MinNUL' is set to zero, while 'MaxNUL' is taken as the number of unused links obtained from the initial solution. In the subsequent steps, membership values for each objective are calculated by using Equations (7), (8), and (9).

Stopping criterion

The algorithm is stopped when a maximum number of iterations has been reached.

6.2 Proposed Fuzzy Simulated Evolution Algorithm

This section details the fuzzy simulated evolution, FSimE, algorithm proposed for the OSPFWS problem. The SimE algorithm operates on a single solution, but unlike SA, navigates the search space by employing compound moves. The algorithm repetitively iterates between the evaluation, selection, and allocation functions before a predefined stopping criterion is reached.

Evaluation

For the OSPFWS problem, each element is a weight on a link, whose goodness needs to evaluated. The goodness function is one of the key factors that affects the performance of the FSimE algorithm, and therefore should be carefully designed. This paper uses the goodness function defined by Sqalli *et al.* [18] for both cost functions (SqalliCF and FuzzyCF). The goodness function is

$$g_i = \begin{cases} 1 - u_i & \text{for } MU \leq 1 \\ 1 - u_i/MU + u_i/MU^2 & \text{for } MU > 1 \end{cases} \quad (12)$$

where u_i represents the utilization on link i and MU refers to the maximum utilization. To illustrate how the above goodness function works, consider the following example: consider two links e_1 and e_2 of a particular solution. Let the maximum utilization be 0.9. Assume that the utilizations on links e_1 and e_2 are 0.6 and 0.1, respectively. By substituting the above values in the goodness function in Equation (12), $g_{e_1} = 0.4$ and $g_{e_2} = 0.9$ are obtained. In the next iteration, e_1 is more probable to be selected for replacement than e_2. This is because the goodness of e_1 is worse than that of e_2. Now, consider the maximum utilization to be 1.8, and let the utilization on links e_1 and e_2 be 0.7 and 1.4, respectively. From Equation (12), $g_{e_1} = 0.827$ and $g_{e_2} = 0.655$. In the next iteration, e_2 is more probable to be selected for replacement than e_1.

Selection

In the selection phase, for each link, i, a random number is sampled from a uniform distribution in the range [0,1]. If this random number is larger than $g_i + B$, the corresponding weight is selected for allocation.

The bias B is used to control the size of the set of selected weights. A low value of B increases the number of elements selected in each iteration, thus allowing the algorithm to explore more. This may lead to high quality solutions, but at the expense of higher computational effort. A high value of B inflates the goodness of each element. This may result in a reduced number of elements selected for reallocation. Consequently, the execution time of the algorithm is reduced, but at the risk of premature convergence to a sub-optimal (or local optimal) solution.

Since it is computationally expensive to find the best bias value by a process of trial-and-error, different approaches have been proposed in the literature [30][45] to use a dynamic bias, instead of a user-defined static bias. The approach by Sait *et al.* [30] calculates the bias based on the quality of the current solution and changes every iteration. The corresponding dynamic bias is given by

$$B(t) = 1 - G(t) \tag{13}$$

where $B(t)$ is the bias in iteration t and $G(t)$ is the average goodness of all the elements at the start of that iteration. The average goodness of elements is a measure of how many "good" elements are present in the solution. For a detailed analysis of dynamic bias, the interested reader is referred to Sait *et al.* [30].

Allocation

During the allocation stage of the algorithm, the selected weights are removed from the solution one at a time. For each removed weight, new weights are tried in such a way that they result in an overall better solution. For the OSPFWS problem, weights are in the range [1,20]. The following allocation scheme was adopted. For all the iterations, a weight window of value 4 is kept. Therefore, if weight 6 was selected for replacement, then weight values 4, 5, 7, and 8 are tried. This results in a beam search and is done to have less disturbance in the solutions in each iteration.

7 Results and Discussion

This paper uses the test cases proposed by Fortz and Thorup [3]. Table 1 shows the characteristics of the test cases. For each test case, the table lists its network type, the number of nodes, and the number of links. The *2-level hierarchical networks* are generated using the GT-ITM generator [46], based on the model of Calvert [47] and Zegura [48]. 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*[1], 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* [2] on the network. The demands generated by Fortz and Thorup assign higher demands to closely located node pairs. For further details on the assignment process of generated

[1] Waxman graphs are frequently chosen in simulations as topologies resembling communications networks. Waxman graphs are named after Bernard M. Waxman.

[2] A hot spot is a network point or router having heavy incoming and outgoing traffic.

demands, the interested reader is referred to Fortz and Thorup [6].

7.1 Purpose and Outline

The following discussion will focus on the experimental procedure adopted in this paper and the type of experiments performed. The purpose of the discussion is mainly to asses the performance of the two cost functions, namely, SqalliCF and FuzzyCF, as well as the mutual comparison of the fuzzy simulated annealing and fuzzy simulated evolution algorithms.

7.2 Experimental Procedure

For each test case, 30 independent runs were executed, and the average of the best solutions found in each run was reported, together with the standard deviation. The average runtime over the 30 runs for each test case was also recorded. Furthermore, the Wilcoxon rank-sum test [49] was used to validate the significance of the results. The associated ranks and corresponding p-values were also provided. A confidence level of 95% was used.

Four sets of experiments were conducted. The first set of experiment compared the two cost functions (i.e. SqalliCF and FuzzyCF) with regard to the final values (average of 30 runs) of each of the three objectives, using SA. The second set of experiments did the same comparison, but using SimE. The third set of experiments compared the performance of SA and SimE with regard to FuzzyCF only (since FuzzyCF is proposed in this paper and therefore, the focus of the paper is on FuzzyCF). Finally, the fourth set of experiments focussed on the comparison of fuzzy SimE with NSGA-II. Detailed results and analysis of each set of experiments are given below.

7.3 Comparison of SqalliCF and FuzzyCF using SA

A value of $\alpha = 0.965$ was used by Sqalli *et al* [1]. The same value was used in this paper for both SqalliCF and FuzzyCF. However, experimentation was done with values of $M = 20$ and $M = 30$ using test cases h50N212a and h100N360a. As observed from Table 2, $M = 20$ resulted in better cost values for both SqalliCF (a lower value is desired) and FuzzyCF (a higher value is desired). Statistical testing suggested that for FuzzyCF, results produced by $M = 20$ were better than those by $M = 30$. For SqalliCF, statistical testing showed that there was no significant difference between the results produced by $M = 20$ and $M = 30$. However, since $M = 20$ requires less perturbations, and thus results in lesser execution time, $M = 20$ is preferred over $M = 30$. Therefore, all experiments were conducted using $M = 20$ for both FuzzyCF and SqalliCF. Furthermore, 5000 iterations were used for each run.

Tables 3 and 4 respectively summarize the results obtained for SA using SqalliCF and FuzzyCF, and the percentage improvement achieved by FuzzyCF when compared to SqalliCF. The results in both tables are displayed with respect to the three design objectives, i.e. maximum utilization (MU), number of congested links (NOC), and number of unutilized links (NUL). Furthermore, average runtime for each test case with respect to the two cost functions is also provided.

From Table 3, it is observed that SqalliCF was able to generate lower levels of MU as compared to FuzzyCF for all test cases. More specifically, MU for SqalliCF was mainly in the range 1.23 to 1.93, with the exception of r50N245a (having an MU level of 2.24). For FuzzyCF, MU was mainly in the range 1.41 to 3.58. The difference between MU levels of the two approaches were statistically evaluated for significance in terms of percentage improvements as shown in Table 4. The results in column two of the table suggest that, for the objective MU, the results obtained by UAO were of inferior quality than those of SqalliCF, as confirmed by the Wilcoxon test.

With respect to the second objective (NOC), the results were somewhat mixed as shown in Table 3. For all Waxman graphs as well as test case h100N360a, SqalliCF resulted in lower number of congested links, while for the remaining test cases, the FuzzyCF cost function showed better performance. This was confirmed by values in Table 4 which showed statistically better results for FuzzyCF than SqalliCF for four test cases (h100N280a, h50N148a, r50N228a, and r50N245a), while SqalliCF had statistically better performance for four other cases (h100N360a, w100N391a, w100N476a, and w50N230a). For the remaining four cases, both FuzzyCF and SqalliCF had the same quality of results.

Table 3 reflects that, for the third objective (i.e. NUL), the dominant trend was that FuzzyCF performed better than SqalliCF for over 80% (i.e. 10 out of 12) test cases. As seen in Table 4, results obtained by FuzzyCF were significantly better than SqalliCF except for test case w100N391a, while for r100N503a, results were of the same quality.

With regard to the execution time, it is observed from Table 3 that the average execution time per run for FuzzyCF was lesser than that of SqalliCF.

Based on the above observations and analysis, it can be suggested that overall, FuzzyCF performed better than SqalliCF.

7.4 Comparison of SqalliCF and FuzzyCF for SimE

As mentioned above, the bias B can have a significant impact on the performance of the SimE algorithm.

Therefore, it is important to find the most appropriate value of the bias. For the SqalliCF cost function, a bias value of -0.02 was used as reported by Sait *et al.* [18]. To find a suitable bias value for FuzzyCF, experiments were done with static bias values of -0.1, 0, 0.1, 0.2, 0.3, as well as with a dynamic bias (see Equation (13)) using two test cases of different complexities, one with 50 nodes and the other with 100 nodes. These test cases were h100N360a and h50N212a. The average cost over 30 runs for each bias value is shown in Table 5. It is observed from this table, that a bias value of -0.1 produced the best results (i.e. highest value of FuzzyCF) for both test cases. Furthermore, a dynamic bias was not able to produce high quality results. Therefore, for all experiments involving SimE with FuzzyCF, a bias value of -0.1 was used.

The results for SimE were more or less the same as those obtained with SA as far as the MU and NUL objectives are concerned (refer to Tables 6 and 7). The average value of maximum utilization (MU) for all test cases was lower for SqalliCF than that of FuzzyCF. From column three of Table 6, note that MU using SqalliCF was in the range 1.23 to 1.44, with the exception of r50N245a (with MU level of 2.14). For FuzzyCF, the MU was mainly in the range 1.41 to 1.72, with two exceptions of r100n503a and r50N245a having MU values of 2.19 and 2.60 respectively. Results of statistical testing on percentage improvement (column two of Table 7) suggest that FuzzyCF produced inferior results compared to SqalliCF for all test cases.

As far as the second objective (NOC) is concerned, the results in Table 6 show superior performance by FuzzyCF for seven cases. This is confirmed by statistical testing and results in Table 7, where the FuzzyCF function showed statistically better results than SqalliCF for six test cases, while for two test cases (h100N360a and h50N148a), the results were of the same quality. Furthermore, SqalliCF showed statistically better performance for the remaining cases (r100N503a, w100N391a, w100N476a, and w50N230a).

For the NUL objective, FuzzyCF had a lower number of unutilized links (which is desired) than SqalliCF for all test cases, as depicted in Table 6. As suggested by values in Table 7, all results obtained by FuzzyCF were significantly better than SqalliCF.

As far as the execution time is concerned, the results depict more or less the same trend as observed for SA. It can be seen from Table 6 that the average execution time per run for FuzzyCF was lesser than that of SqalliCF.

In view of the above results and analysis, the overall assessment is that FuzzyCF performed better than SqalliCF.

7.5 Comparison of Simulated Evolution and Simulated Annealing Algorithms

A comparative study of SimE and SA with respect to the three design objectives using the SqalliCF and FuzzyCF functions was also performed. Figure 8 shows different plots with respect to maximum utilization, number of congested links, and number of unused links. Figure 8(a) shows that the utilization level for SA is, in general, higher than that of SimE using SqalliCF. This trend is more prominent when SimE and SA are compared using the FuzzyCF operator, as illustrated in Figure 8(b).

With regards to NOC, Figure 8(c) shows that, although SimE and SA have similar performance for most test cases using SqalliCF, there are instances where SimE has less congested links than SA. Therefore, it can be suggested that SimE did relatively better than SA. The better performance of SimE is more prominently visible in Figure 8(d), where SimE has less congested links than SA using FuzzyCF.

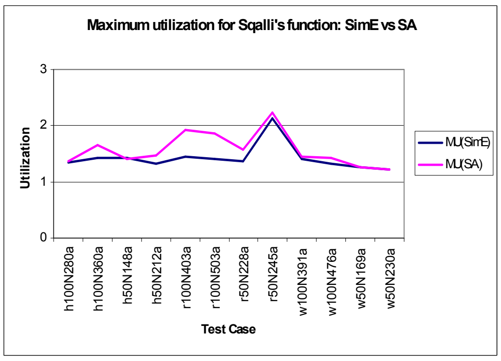
Finally, for the third objective, NUL, both SimE and SA have similar performance using SqalliCF, with some instances showing better performance by SimE, as observed in Figure 8(e). A similar trend is observed in Figure 8(f) with respect to the FuzzyCF function.

With regard to the algorithm execution time, it can be observed from Tables 3 and 6 that with respect to the SqalliCF, the average execution time per run was slightly higher for SimE than that of SA. A similar observation is made with regard to FuzzyCF, where SimE again took slightly more execution time than SA.

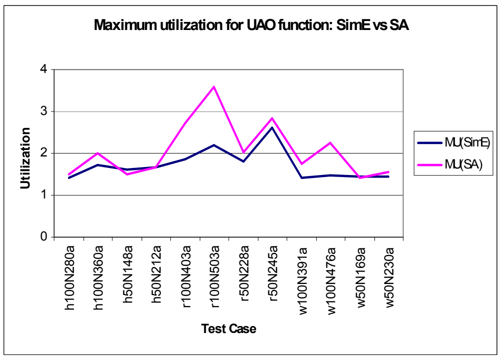
The overall moderately better performance of SimE compared to SA is attributed to the fact that the search in SA is carried out blindly, and moves are done randomly. This may result in replacing an optimally placed weight on a link with a weight resulting in a low quality solution. However, in SimE, perturbations to a solution are done on the on basis of goodness, i.e. moves are done intelligently, rather than blindly. Thus, elements with a higher quality have a lower probability of getting removed, while elements with low quality are more prone to being replaced. Hence, SimE performs more intelligently than SA, resulting in higher quality solutions.

7.6 Comparison of Fuzzy SimE and NSGA-II

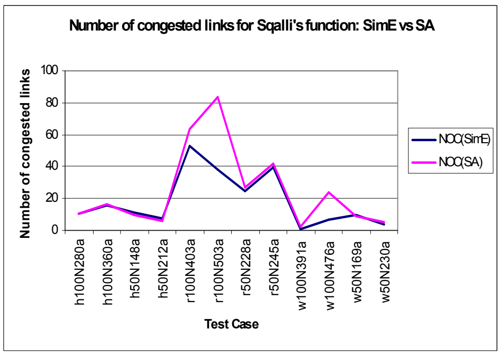
As observed from the above discussion, fuzzy SimE performed relatively better than fuzzy SA. In order to further establish its effectiveness for the problem in hand, the fuzzy SA algorithm was compared to a well-known MOGA, namely, the non-dominated sorting genetic algorithm II (NSGA-II) [50]. Literature has reported that



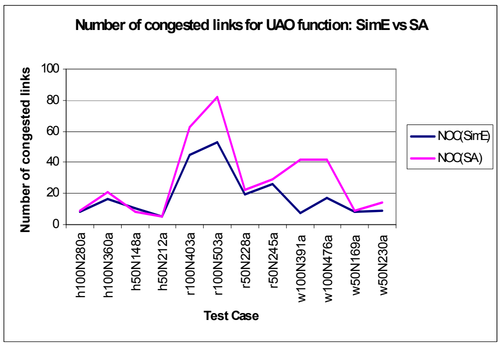
(a)



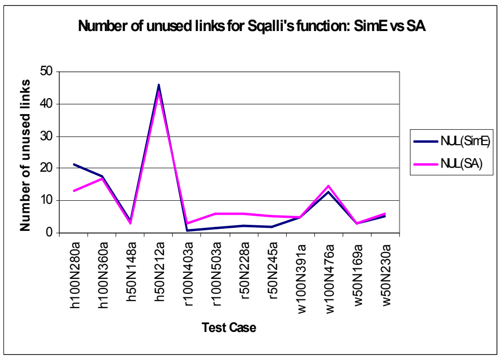
(b)



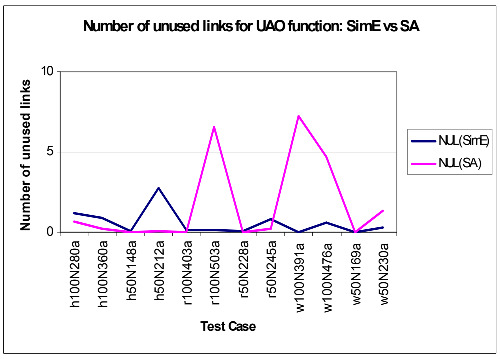
(c)



(d)



(e)



(f)

Fig. 8 Comparison of SimE and SA using SqalliCF and FuzzyCF functions. (a) Maximum utilization using SA (b)Number of congested links using SA (c) Number of unused links using SA (d) Maximum utilization using SimE (e) Number of congested links using SimE (f) Number of unused links using SimE.

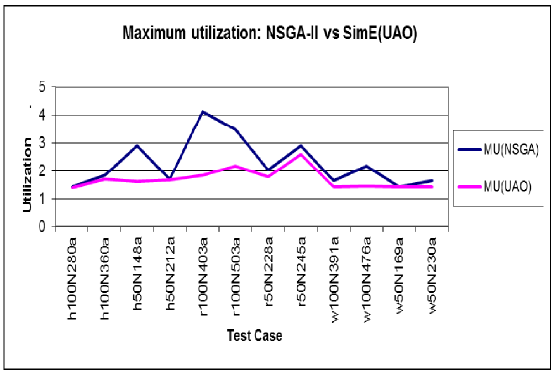
NSGA-II has the capability to converge to the global Pareto-optimal front as well as to maintain the diversity of population on the Pareto-optimal front [50,51]. Furthermore, NSGA-II has lesser computational complexity than other multi-objective evolutionary algorithms. Further details on NSGA-II can be found in [50,51].

The NSGA-II was adapted to address the OSPF weight setting problem and comparison of fuzzy SimE and NSGA-II was done with respect to the three design objectives. Figure 9 depicts performance plots for the two algorithms with respect to maximum utilization, number of congested links, and number of unused links. In Figure 9(a), it is observed that, in general, the maximum utilization level for NSGA is somewhat comparable to that of fuzzy SimE. In eight of the twelve test cases, both NSGA-II and fuzzy SimE were able to achieve the same or nearly same utilization level. However, there are four instances (h50n148a, r100N403a, r100n503a, and w100n476a) where NSGA-II ended up with higher utilization levels than fuzzy SimE. With these observations, it can be claimed that fuzzy SimE was able to demonstrate better performance than NSGA-II as far as the MU objective is concerned.

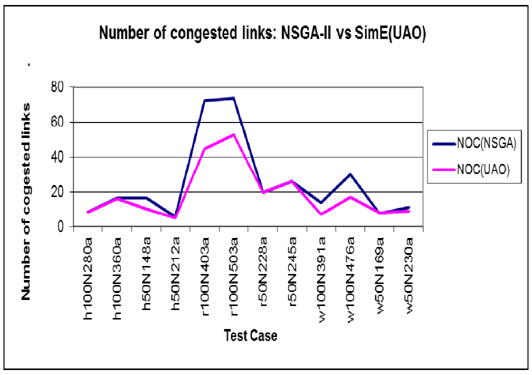
With regards to NOC, Figure 9(b) shows that fuzzy SimE and NSGA achieved exactly the same level of performance for seven test cases. However, there are five instances (h50n148a, r100N403a, r100n503a, w100N391a, and w100n476a) where fuzzy SimE was able to reach lesser number of congested links than NSGA. Therefore, it can be suggested that fuzzy SimE did slightly better than NSGA-II.

Finally, for the NUL objective, NSGA-II performed better than fuzzy SimE for six cases, and achieved almost the same level of results for four test cases as observed from Figure 9 (c). The figure also shows the two exceptions (r100N403a and r100503a) where fuzzy SimE showed better performance than NSGA-II. In general, it can be comfortably claimed that NSGA-II demonstrated superior performance than fuzzy SimE for the NUL objective.

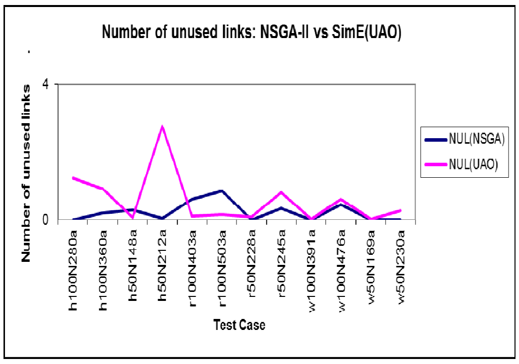
From the above discussion, it turns out that NSGA-II achieved slightly inferior results than fuzzy SimE for the objectives of maximum utilization and number of congested links. However, this inferior performance of NSGA-II was somewhat compensated by its superior performance with regard to the objective of unused links. Therefore, an overall trend is that fuzzy SimE showed a slightly better performance than NSGA-II. The results thus indicate that fuzzy SimE was more efficient in performing search in the areas that had better solutions with the regard to the MU and NOC objectives, whereas NSGA-II was stronger in performing search focussed on obtaining solutions with lower number of unused links.



(a)



(b)



(c)

Fig. 9 Comparison of fuzzy SimE (with UAO) and NSGA-II (a) Maximum utilization (b) Number of congested links, and (c) Number of unused links.

8 Conclusion

The open shortest path first (OSPF) weight setting problem is a complex optimization problem involving multiple conflicting objectives, namely, maximum utilization, number of congested links, and number of unused links. This paper presented two optimization approaches, namely, fuzzy simulated evolution and fuzzy simulated annealing, to efficiently solve the multi-objective OSPF weight setting problem. Fuzzy logic was used to aggregate the three design objectives into a single optimization function using the Unified And-OR (UAO) operator. Results from the UAO based function were compared with results of the Sqalli cost function. Results suggested that overall, the UAO function produced moderately better results than Sqalli's function when used with the fuzzy simulated annealing and fuzzy simulated evolution algorithms. Analysis also suggested that the fuzzy simulated evolution algorithm demonstrated better performance than fuzzy simulated annealing algorithm. Furthermore, a comparison of fuzzy SimE with NSGA-II suggested that fuzzy SimE performed slightly better than NSGA-II. The plan for future work is to focus on applying swarm intelligence algorithms, such as particle swarm optimization and ant colony optimization, to the problem studied in this paper.

9 Appendix

9.1 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

9.2 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.

References

1. 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.
2. 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.
3. B. Fortz and M. Thorup. Internet Traffic Engineering by Optimizing OSPF Weights. *IEEE Conference on Computer Communications(INFOCOM)*, pages 519–528, 2000.
4. J. F. Kurose and K.W. Ross. *Computer Networking: A Top-Down Approach Featuring the Internet.* Prentice Hall Series, 2002.
5. E. W. Dijkstra. A Node on Two Problems in Connection of Graphs. *Numerical Mathematics*, 1959.
6. B. Fortz and M. Thorup. Increasing Internet Capacity using Local Search. *Technical Report IS-MG*, 2000.
7. Kirkpatrick S, Gelatt C, and Vecchi M. Optimization by Simulated Annealing. *Science*, pages 498–516, 1983.
8. 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.
9. M. Rodrigues and K. G. Ramakrishnan. Optimal Routing in Data Networks. *Presentation at International Telecommunication Symposium (ITS)*, 1994.
10. 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.
11. F. Glover and M. Laguna. *Tabu Search.* Kluwer Academic Publishers, 1997.

12. 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.
13. G. Ramalingam and T. Reps. An Incremental Algorithm for a Generalization of the Shortest Path Problem. *Journal of Algorithms*, pages 267–305, 1996.
14. B. Fortz. Combinatorial Optimization and Telecommunications. *http://www.poms.ucl.ac.be/staff/bf/en/COCom-5.pdf.*
15. J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 1975.
16. P. Laarhoven and E. Aarts. *Simulated Annealing: Theory and Applications.* Kluwer Academic, Norwell, Massachusetts, 1987.
17. 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.
18. 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.
19. M. Houssaini Sqalli, S. Mohammed Sait, and S. Asadullah. Minimizing the Number of Congested Links in OSPF Routing. *ATNAC*, December 2008.
20. 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.
21. 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.
22. S. Mohammed Sait and H. Youssef. *Iterative Computer Algorithms and their Application to Engineering.* IEEE Computer Society Press, December 1999.
23. 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.
24. R. Kling and P. Banerjee. ESP: Placement by Simulated Evolution. *IEEE Transactions on Computer-Aided Design*, pages 245–255, March 1989.
25. S. A. Khan and A. P. Engelbrecht. Fuzzy Hybrid Simulated Annealing Algorithms for Topology Design of Switched Local Area Networks. *Soft Computing*, 3(1):45–61, 2009.
26. S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb. A simulated annealing-based multiobjective optimization algorithm: Amosa. *IEEE Trans. Evolutionary Computation*, 12(3):269–283, 2008.
27. B. Suman, N. Hoda, and S. Jha. Orthogonal simulated annealing for multiobjective optimization. *Computers & Chemical Engineering*, 34(10):1618–1631, 2010.
28. B. Suman. Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem. *Computers & Chemical Engineering*, 28(9):1849–1871, 2004.
29. 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.
30. S. Sait, H. Youssef, and A. Hussain. Fuzzy simulated evolution algorithm for multiobjective optimization of VLSI placement. In *IEEE Congress on Evolutionary Computation, Washington*, pages 91–97, 1999.
31. S. Sait, A. Zaidi, and M. Ali. Multiobjective vlsi cell placement using distributed simulated evolution algorithm. In *ISCAS 2005*, pages 6226–6229, 2005.

32. B. Fortz, J. Rexford, and M. Thorup. Traffic Engineering with Traditional IP Routing Protocols. *IEEE Communications Magazine*, pages 118–124, 2002.
33. L. A. Zadeh. Fuzzy Sets. *Information Control*, 8:338–353, 1965.
34. L. A. Zadeh. The Concept of a Linguistic Variable and its Application to Approximate Reasoning. *Information Sciences*, 8:199–249, 1975.
35. R. Yager. On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decision-making. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):183–190, Jan 1988.
36. H. Li and V. Yen. *Fuzzy Sets and Fuzzy Decision-Making*. CRC Press, USA, 1995.
37. H. Hamacher. Ueber Logische Verknupfungen Unschalfer Aussagen und deren Zugehoerige Bewertungs-funktione. *Progress in Cybernetics and Systems Research*, 3:276–288, 1978.
38. M. Frank. On the Simultaneous Associativity of $F(x, y)$ and $x+y-F(x, y)$. *Aequationes Mathematicae*, 19:194–226, 1979.
39. 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.
40. D. Dubois and H. Prade. Operations in Fuzzy-valued Logic. *Information and Control*, 43:224–240, 1979.
41. Cho H, Oh S, and Choi D. A new Evolutionary Programming Approach Based on Simulated Annealing with Local Cooling Schedule. *Proceedings of the IEEE World Congress on Computational Intelligence*, pages 598–602, 1998.
42. Jeon Y, Kim JC, Kim JO, Shin J, and Lee K. An Efficient Simulated Annealing Algorithm for Network Reconfiguration in Large-Scale Distribution Systems. *IEEE Transactions Power Delivery*, pages 1070–1078, 2002.
43. MatSuba I. Optimal Simulated Annealing Method and its Application to Combinatorial Problems. *Proceedings of the International Joint Conference on Neural Networks*, pages 541–546, 1989.
44. Perttunen C. Nonparametric Cooling Schedules in Simulated Annealing Using the Normal Score Transformations. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pages 609–612, 1991.
45. S. A. Khan. *Design and Analysis of Evolutionary and Swarm Intelligence Techniques for Topology Design of Distributed Local Area Networks.* PhD Thesis, University of Pretoria, 2009.
46. 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.
47. K. Calvert, M. Doar, and E. W. Zegura. Modeling Internet Toplogy. *IEEE Communications Magazine*, (35):160–163, 1997.
48. 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.
49. W. Hines and D. Montgomery. *Probability and Statistics in Engineering and Management Science, 3rd Ed.* John Wiley & Sons, 1990.
50. 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.
51. K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. pages 849–858. Springer, 2000.

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 Results of average cost for different values of M for SqalliCF and FuzzyCF

Test Case	SqalliCF		FuzzyCF	
	$M = 20$	$M = 30$	$M = 20$	$M = 30$
h50N212a	2.498	3.395	0.461	0.261
h100N360a	3.003	4.994	0.308	0.217

Table 3 MU, NOC, NUL, and average execution time corresponding to the two cost functions using SA

Test case	Traffic Demand (in bytes)	SqalliCF				FuzzyCF (UAO)			
		MU	NOC	NUL	Time (sec)	MU	NOC	NUL	Time (sec)
h100N280a	4605	1.36± 0.017	10.17± 2.422	13.13± 3.181	1177	1.50± 0.352	8.93± 1.388	0.70± 3.313	859.1
h100N360a	12407	1.66± 0.157	16.20± 3.067	16.93± 4.401	1091.1	2.01± 0.531	21.03± 6.300	0.20± 0.484	796.5
h50N148a	4928	1.40± 0.058	9.63± 2.251	3.07± 1.460	135.2	1.51± 0.100	8.40± 1.670	0.00± 0.000	98.2
h50N212a	3363	1.47± 0.277	5.63± 1.670	43.70± 5.977	145.7	1.68± 0.080	5.17± 0.791	0.07± 0.254	102.1
r100N403a	70000	1.93± 0.184	63.50± 5.124	3.00± 1.681	1335.5	2.72± 0.576	62.60± 6.579	0.03± 0.183	1017.2
r100N503a	100594	1.86± 0.143	83.60± 6.184	5.87± 2.713	1390.4	3.58± 0.416	82.33± 26.212	6.57± 16.079	1069.8
r50N228a	42281	1.58± 0.154	26.97± 3.347	5.80± 2.441	160.3	2.02± 0.134	22.03± 2.251	0.03± 0.183	116.1
r50N245a	53562	2.24± 0.276	41.83± 4.913	5.10± 2.056	159.1	2.83± 0.239	28.77± 2.622	0.23± 0.504	117.2
w100N391a	48474	1.44± 0.046	2.33± 1.647	4.80± 1.954	1355.5	1.75± 0.708	42.10± 19.921	7.23± 6.185	1029.2
w100N476a	63493	1.42± 0.043	23.83± 6.923	14.73± 3.433	1383.3	2.24± 0.219	41.70± 15.647	4.73± 11.694	1048.2
w50N169a	25411	1.27± 0.021	8.60± 1.792	2.93± 1.552	153.7	1.41± 0.086	8.80± 1.808	0.00± 0.000	110.7
w50N230a	39447	1.23± 0.017	4.90± 1.493	6.13± 2.674	158.6	1.55± 0.151	14.40± 19.08	1.37± 4.537	113.8

Table 4 Percentage improvement achieved by FuzzyCF compared to SqalliCF using SA. Statistically significant improvements are in boldface

Test Case	MU				NOC				NUL			
	% diff	p-value	rank Fuzzy	rank Sqalli	% diff	p-value	rank Fuzzy	rank Sqalli	% diff	p-value	rank Fuzzy	rank Sqalli
h100N280a	**-10.29**	0.022	473	443	**12.19**	0.019	119	156	**94.67**	0.013	44	275
h100N360a	**-21.08**	0.001	1233	555	**-29.81**	0.027	335	216	**98.82**	0.011	36	329
h50N148a	**-7.86**	0.031	1191	639	**12.77**	0.019	102	135	**100.00**	0.010	30	122
h50N212a	**-14.29**	0.022	1107	723	8.17	0.172	125	139	**99.84**	0.010	32	363
r100N403a	**-40.93**	0.003	1355	475	1.42	0.557	335	367	**99.00**	0.027	31	118
r100N503a	**-92.47**	0.001	1365	465	1.52	0.798	293	506	-11.93	0.815	96	206
r50N228a	**-27.85**	0.021	1345	485	**18.32**	0.041	151	296	**99.48**	0.009	31	204
r50N245a	**-26.34**	0.034	1309	521	**31.22**	0.035	143	494	**95.49**	0.020	37	183
w100N391a	**-21.53**	0.021	691	334	**-1706.87**	0.000	538	69	**-50.63**	0.044	247	174
w100N476a	**-57.75**	0.009	1365	465	**-74.99**	0.008	611	277	**67.89**	0.032	104	330
w50N169a	**-11.02**	0.033	991	240	-2.33	0.669	144	138	**100.00**	0.010	30	116
w50N230a	**-26.02**	0.017	972	181	**-193.88**	0.001	289	87	**77.65**	0.000	61	211

Table 5 Results of average cost for different bias values using FuzzyCF

Test Case	Bias	Average Cost
h50N212a	-0.1	0.466
	0.0	0.395
	0.1	0.287
	0.2	0.266
	0.3	0.220
	- 0.02	0.415
	Dynamic	0.293
h100N360a	-0.1	0.513
	0.0	0.429
	0.1	0.348
	0.2	0.293
	0.3	0.166
	- 0.02	0.370
	Dynamic	0.357

Table 6 MU, NOC, NUL, and average execution time corresponding to two cost functions using SimE

Test case	Traffic Demand (in bytes)	SqalliCF				FuzzyCF (UAO)			
		MU	NOC	NUL	Time (sec)	MU	NOC	NUL	Time (sec)
h100N280a	4605	1.34± 0.000	10.73± 2.449	21.33± 3.078	1220.4	1.41± 0.058	8.50± 1.432	1.23± 0.898	875.9
h100N360a	12407	1.42± 0.088	15.63± 2.189	17.67± 3.708	1189.6	1.72± 0.092	16.13± 2.240	0.93± 1.015	826.6
h50N148a	4928	1.42± 0.051	10.93± 2.518	3.67± 1.668	148.1	1.62± 0.119	10.37± 2.025	0.07± 0.254	128.6
h50N212a	3363	1.32± 0.096	7.60± 1.714	46.07± 5.705	151	1.67± 0.102	4.93± 0.740	2.77± 1.924	137.5
r100N403a	70000	1.44± 0.064	53.13± 4.562	0.83± 0.648	1597.9	1.86± 0.074	44.73± 2.490	0.13± 0.346	1588.5
r100N503a	100594	1.41± 0.059	37.77± 5.283	1.63± 0.890	1479.2	2.19± 0.166	52.80± 2.809	0.17± 0.379	1334.1
r50N228a	42281	1.36± 0.054	24.83± 3.130	2.20± 1.808	160	1.80± 0.121	19.77± 1.305	0.10± 0.305	152.2
r50N245a	53562	2.14± 0.182	39.20± 3.872	2.03± 0.999	167.2	2.60± 0.193	26.20± 1.972	0.80± 0.887	114.4
w100N391a	48474	1.41± 0.006	1.10± 0.305	5.03± 1.829	1371	1.42± 0.033	7.17± 2.692	0.03± 0.183	1132.3
w100N476a	63493	1.32± 0.003	7.07± 1.143	12.63± 1.712	1374.6	1.46± 0.070	17.07± 2.363	0.60± 0.814	1174.9
w50N169a	25411	1.26± 0.016	9.37± 2.205	2.80± 1.375	178.4	1.44± 0.078	8.37± 1.189	0.03± 0.183	134.8
w50N230a	39447	1.23± 0.007	3.67± 0.959	5.27± 1.741	189.1	1.44± 0.088	9.13± 1.570	0.27± 0.583	112.8

Table 7 Percentage improvement achieved by FuzzyCF compared to SqalliCF for SimE. Statistically significant improvements are in boldface.

Test Case	MU				NOC				NUL			
	% diff	p-value	rank Fuzzy	rank Sqalli	% diff	p-value	rank Fuzzy	rank Sqalli	% diff	p-value	rank Fuzzy	rank Sqalli
h100N280a	**-5.22**	0.032	381	30	**20.78**	0.040	135	202	**94.23**	0.019	67	281
h100N360a	**-21.13**	0.027	1170	381	-3.20	0.385	183	172	**94.74**	0.018	58	350
h50N148a	**-14.08**	0.020	1308	522	5.12	0.341	131	148	**98.09**	0.020	32	141
h50N212a	**-26.52**	0.022	1355	475	**35.13**	0.033	88	168	**93.99**	0.011	113	530
r100N403a	**-29.17**	0.020	1365	465	**15.81**	0.031	143	358	**84.34**	0.010	34	55
r100N503a	**-55.32**	0.016	1365	465	**-39.79**	0.027	680	292	**89.57**	0.012	35	79
r50N228a	**-32.35**	0.019	1365	465	**20.38**	0.019	113	246	**95.45**	0.034	33	93
r50N245a	**-21.50**	0.011	1328	502	**33.16**	0.020	156	420	**60.59**	0.023	54	91
w100N391a	**-0.71**	0.045	157	133	**-551.82**	0.000	176	33	**99.40**	0.011	31	180
w100N476a	**-10.61**	0.022	615	40	**-141.44**	0.001	332	62	**95.25**	0.025	48	229
w50N169a	**-14.29**	0.020	726	134	**10.67**	0.033	101	131	**98.93**	0.019	31	114
w50N230a	**-17.07**	0.015	735	104	**-148.77**	0.001	214	50	**94.88**	0.021	38	188