Chapter 9

Fuzzy Particle Swarm Optimization for DLAN Topology Design

This chapter presents another swarm intelligence algorithm, namely, the fuzzy particle swarm optimization algorithm. The original particle swarm algorithm was adapted to address the multi-objective aspects of the DLAN topology design using fuzzy logic. The chapter first discusses the main features of the fuzzy PSO algorithm. This is followed by empirical results to evaluate the performance of the fuzzy PSO algorithm with respect to different parameters of the algorithm.

9.1 Fuzzy Particle Swarm Optimization Algorithm

Particle swarm optimization (PSO) was discussed in Chapter 2. Although PSO has been applied to a number of problems, its effectiveness in the multi-objective

domain has to be explored further. The development of a multi-objective PSO for the DLAN topology design problem is one step towards the assessment of the performance of PSO in multi-objective optimization with application to a real-world design problem. Therefore, the focus of this chapter is not to compare the variants and alterations proposed for PSO by many researchers, but rather the development of a fuzzy logic based multi-objective PSO, and a preliminary analysis of the fuzzy PSO with respect to the OWA and UAO operators.

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

9.1.1 Particle Position and Velocity Representation

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

be used. This representation scheme is described below.

A position will be the set

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

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

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

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

Example 1: Consider a simple network of 6 nodes as given in Figure 9.1. The topology in this figure represents a possible configuration at time t, and thus represents a solution (i.e. particle). According to the network configuration in Figure 9.1, the current solution is given as

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

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

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

9.1.2 Velocity Update

The velocity of particle i is updated using

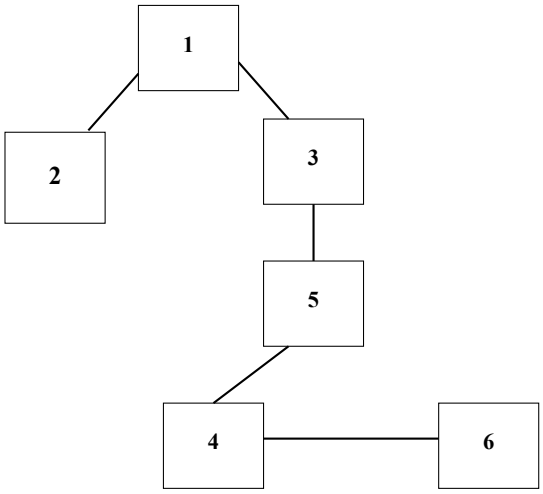


Figure 9.1: Network topology for PSO example

$$\mathbf{V}_i(t+1) = w \otimes \mathbf{V}_i(t) \oplus c_1 r_1(t) \otimes [\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)] \oplus c_2 r_2(t) \otimes [\mathbf{P}_g(t) \oslash \mathbf{X}_i(t)] \quad (9.1)$$

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

In Equation (9.1), the operator $\otimes$ is implemented as follows: the number of elements to be selected are determined as $\lfloor \mathrm{w} \times |\mathbf{V}_i(t)| \rfloor$. Then, the result will be the above number of elements randomly selected from $\mathbf{V}_i(t)$. The same is approach is applicable to other factors where the operator $\otimes$ is used.

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

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

$$c_1 r_1(t) \otimes [\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)] = \lfloor c_1 r_1 \times |\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)| \rfloor \tag{9.2}$$

where $|\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)|$ represents the cardinality of the set. The result of Equation (9.2) indicates the number of elements that are randomly selected from the set $\mathbf{P}_i(t) \oslash \mathbf{X}_i(t)$; $c_2 r_2(t) \otimes [\mathbf{P}_g(t) \oslash \mathbf{X}_i(t)]$ has the same meaning.

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

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

w $= 0.5$

$V_{max} = 2$

$c_1 = c_2 = 0.5$

$r_1 = 0.52$ (randomly generated)

$r_2 = 0.75$ (randomly generated)

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

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

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

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

The inertia weight, w, determines the number of moves that will be randomly selected from $\mathbf{V}_i(t)$ (mentioned in Example 1 above). Since $w = 0.5$, and $|\mathbf{V}_i(t)| =$

3, the number of moves selected is $0.5 \times |\mathbf{V}_i(t)| = 1.5$. Since fractional moves are not possible, the value is truncated to 1.

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

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

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

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

Similarly:

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

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

Putting the above calculations in Equation (9.1) gives $\mathbf{V}_i(t+1)$ containing three elements as

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

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

Hence,

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

9.1.3 Particle Position Update

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

$$\mathbf{X}_i(t+1) = \mathbf{X}_i(t) \boxplus \mathbf{V}_i(t+1) \tag{9.3}$$

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

Example 3: Continuing with Example 2,

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

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

9.1.4 Fitness Evaluation

The fitness (goodness) of a solution is evaluated using either Equation (3.11) or Equation (4.1), as discussed in Chapters 3 and 4.

9.1.5 Initialization

Since PSO is a population-based algorithm, the initialization process consists of generating a set of solutions. This process is exactly the same as discussed in Section 8.1.1 for fuzzy ACO. Algorithm parameters such as inertia weight, velocity clamping, and acceleration constants are also initialized. The goodness of each particle is then evaluated with respect to the reference solution (a predefined initial solution used in the earlier chapters of this thesis).

9.1.6 Particle Activity

It was mentioned in Chapter 2 that PSO has two basic models known as *lbest* PSO and *gbest* PSO. FPSO is based on the *gbest* model (although the *lbest* model could be used as well) to allow comparison with the fuzzy ACO. Recall that, in fuzzy ACO, the elitist ant has a significant effect in guiding the search process towards a certain direction. In other words, the fuzzy ACO is considering a global approach in guidance. Therefore, to match this global approach to the best possible extent, the *gbest* model has been adopted, where the global component is provided by the overall best particle.

Once the initial set of solutions is generated, the global best particle is chosen on the basis of the goodness value calculated in the initialization phase. Also, at this stage, each particle's current position is its best position. In the following iterations, each particle updates its position based on information provided by the particle's immediate previous position and by the alterations (moves) performed on the particle through the velocity update vector, as explained in Section 9.1.3. The velocity of a particle is updated on the basis of moves performed on the particle in its

immediate previous position, the particle's own best position so far, and the overall best position achieved by any particle in the swarm in any iteration, as described in Section 9.1.2. Moreover, to avoid premature convergence, the global best particle is updated regularly, i.e. as soon as a particle's overall goodness becomes higher than the overall goodness of the global best particle, that new particle is selected as the global best particle, and the search process continues. If no updating is done, then the algorithm will very quickly converge on a solution that might not even be a local minimum.

A 'move' in FPSO is very similar to what has been described for other algorithms in earlier chapters: removing a link and introducing a new one such that the tree is maintained (refer to Example 1 in Section 9.1.1). Then, the constraints are checked to evaluate the feasibility of the performed move. However, the notion of moves in FPSO depends on three factors, namely: moves performed in the immediate previous position of the particle, the structure of the particle's own best position, and the structure of the global best particle. For all these factors, the number of moves performed to get the new position of the particle is governed by parameters such as acceleration coefficients, inertia weight, and velocity clamping. Values of these parameters decide how many moves are required to get the new position of a particle.

9.2 Results and Discussion

The fuzzy PSO was applied to the five test cases. The performance of the algorithm was evaluated with respect to a number of parameters. These parameters are the

inertia weight w, velocity clamping V_{max}, swarm size, and acceleration constants. The parameter values used in the experiments are given in Table 9.1. In these experiments, each instance of the algorithm was run for 100 iterations. Thirty independent runs were executed for each parameter setup, and the average of best solutions found in each run was reported, with the standard deviation. Following default values were used for experiments, unless otherwise specified: number of particles = 20, $V_{max} = 5$, $w = 0.72$, and $c_1 = c_2 = 0.5$.

Table 9.1: Parameter settings for fuzzy PSO used in experiments.

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

9.2.1 Effect of Swarm size

The effect of swarm size was investigated with different number of particles as given in Table 9.1. Other parameters were kept as follows: $V_{max} = 5$, $c_1 = c_2 = 0.5$, and inertia weight $w = 0.72$. Tables 9.2 to 9.6 reflect the effect of number of particles on the quality of solution. Column 1 lists the test case, column 2 gives the overall goodness obtained using the OWA operator, while column 3 provides the corresponding run time, and column 4 lists the percentage difference between the

corresponding number of particles and the best goodness (in boldface). Column 5 reports the overall goodness obtained using the UAO operator, column 6 provides the corresponding run time, and column 7 lists the percentage difference between the corresponding number of particles and the best goodness (in boldface). For example, in Table 9.2, the best overall goodness (in boldface) using OWA is obtained with 30 particles. The overall goodness with different numbers of particles is then compared with the best overall goodness, and the percentage difference is listed. It is evident from Tables 9.2 to 9.6 that the best overall goodness was obtained when the number of particles was relatively high. With OWA, the best results for *n50* and *n40* were obtained with 30 particles, while for *n25* and *n15*, 25 particles resulted in the best solutions. Only in test case *n33* did a moderate number of particles, 20 in this case, demonstrate better performance.

As for UAO, the effect of the large swarm size was even more prominent, where it is noticed that the best results were obtained in all cases with a swarm size of 30. A small deviation from this trend was the case *n33* where 25 particles produced the best overall goodness.

Table 9.2: Effect of swarm size on overall goodness for *n50* with OWA and UAO. Time = Run time (in seconds), % Diff = % Difference. Statistically significant difference is in italics.

Number of particles	Goodness OWA	Time	% Diff	Goodness UAO	Time	% Diff
10	0.251 ±0.074	2057.7	*14.26*	0.333 ±0.005	2151.9	*0.89*
15	0.268 ±0.041	3041.9	6.94	0.334 ±0.004	3510.4	*0.72*
20	0.263 ±0.039	4291.1	*9.02*	0.335 ±0.004	4505.4	0.23
25	0.269 ±0.042	5352.3	6.72	0.335 ±0.002	5704.9	0.43
30	**0.287** ±0.034	6328.1	NA	**0.336** ±0.003	7074.9	NA

Table 9.3: Effect of swarm size on overall goodness for *n40* with OWA and UAO. Time = Run time (in seconds), % Diff = % Difference. Statistically significant difference is in italics.

Number of particles	Goodness OWA	Time	% Diff	Goodness UAO	Time	% Diff
10	0.296 ±0.039	544.9	*14.12*	0.338 ±0.005	545.3	*3.90*
15	0.326 ±0.044	782.5	3.61	0.341 ±0.012	829.5	*2.92*
20	0.318 ±0.036	1132.9	*6.20*	0.342 ±0.009	1081.8	*2.80*
25	0.316 ±0.023	1390.7	*6.78*	0.346 ±0.010	1428.2	1.50
30	**0.337** ±0.026	1736.7	NA	**0.351** ±0.012	1659.2	NA

Table 9.4: Effect of swarm size on overall goodness for *n33* with OWA and UAO. Time = Run time (in seconds), % Diff = % Difference. Statistically significant difference is in italics.

Number of particles	Goodness OWA	Time	% Diff	Goodness UAO	Time	% Diff
10	0.300 ±0.041	215.8	4.17	0.332 ±0.006	210.5	*2.06*
15	0.306 ±0.030	324.7	2.29	0.337 ±0.006	329.3	0.56
20	**0.313** ±0.053	445.3	NA	0.337 ±0.005	455.9	0.58
25	0.312 ±0.031	597.9	0.17	**0.339** ±0.005	560.6	NA
30	0.311 ±0.040	667.3	0.63	0.338 ±0.006	662.5	0.16

Table 9.5: Effect of swarm size on overall goodness for *n25* with OWA and UAO. Time = Run time (in seconds), % Diff = % Difference. Statistically significant difference is in italics.

Number of particles	Goodness OWA	Time	% Diff	Goodness UAO	Time	% Diff
10	0.306 ±0.038	62.6	*11.39*	0.330 ±0.009	58.0	*2.69*
15	0.325 ±0.036	92.1	4.70	0.330 ±0.004	91.5	*2.73*
20	0.333 ±0.031	118.7	2.20	0.335 ±0.005	120.3	1.13
25	**0.340** ±0.030	153.0	NA	0.337 ±0.008	159.3	0.60
30	0.327 ±0.032	187.1	4.17	**0.339** ±0.008	187.0	NA

Table 9.6: Effect of swarm size on overall goodness for *n15* with OWA and UAO. Time = Run time (in seconds), % Diff = % Difference. Statistically significant difference is in italics.

Number of particles	Goodness OWA	Time	% Diff	Goodness UAO	Time	% Diff
10	0.186 ±0.020	13.7	*16.98*	0.332 ±0.002	14.3	0.42
15	0.204 ±0.049	22.1	6.85	0.332 ±0.002	21.2	0.42
20	0.218 ±0.031	29.8	-0.09	0.332 ±0.001	28.7	0.32
25	**0.218** ±0.029	36.5	NA	0.332 ±0.002	35.9	0.42
30	0.216 ±0.024	43.9	0.91	**0.333** ±0.003	43.2	NA

A t-test validation of percentage difference in Tables 9.2 to 9.6 was also performed, and the statistically significant differences are italicized. An important observation in Tables 9.2 to 9.6 is that the lowest level of overall goodness was obtained when the number of particles was lowest. More specifically, having 10 particles resulted in the worst solutions in all cases and with both fuzzy operators. The only exception to this trend was *n15* when UAO was applied. In this instance, all particles from 10 to 25 resulted in the same overall goodness value. The improvement between the highest and the lowest overall goodness using the OWA and UAO operators is given in Tables 9.7 and 9.8 respectively. Table 9.7 shows that the improvement was generally between 10% and 15%, with the exception of *n33* where an improvement of 4.17% was observed. Moreover, t-test validation showed that all improvements, except that for *n33*, were statistically significant. As for UAO, the improvement was generally less than 4%, as shown in Table 9.8, and all improvements (except for *n15*) were statistically significant, as validated by the t-test.

A graphical representation of the results in Tables 9.2 to 9.6 is given in Figure

9.2. This figure shows the effect on overall goodness when the number of particles are varied from 10 to 30. This figure further strengthens the observations, noted above, that in general, increasing the number of particles positively affects the quality of overall goodness of the solution. For example, in Figure 9.2(a), the overall goodness increased with an increase in the number of particles for case *n50*. The trend is more obvious for OWA than for UAO. Similarly, for all other cases, this general trend is observed for both the OWA and UAO operators. Only in the instance of *n25* with OWA (Figure 9.2(d)) did the overall goodness increase up to 25 particles and then dropped with 30 particles.

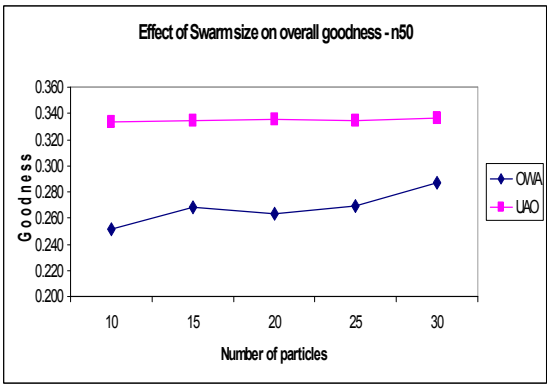
Table 9.7: Results for best and worst average overall goodness and their respective number of particles for OWA. Statistically significant improvement is in italics.

Case	Particles	Max. goodness	Particles	Min. goodness	% improvement
n15	25	0.218 ±0.029	10	0.186 ±0.020	*16.98*
n25	25	0.340 ±0.030	10	0.306 ±0.038	*11.39*
n33	20	0.313 ±0.053	10	0.300 ±0.041	4.17
n40	30	0.337 ±0.026	10	0.296 ±0.039	*14.12*
n50	30	0.287 ±0.034	10	0.251 ±0.074	*14.26*

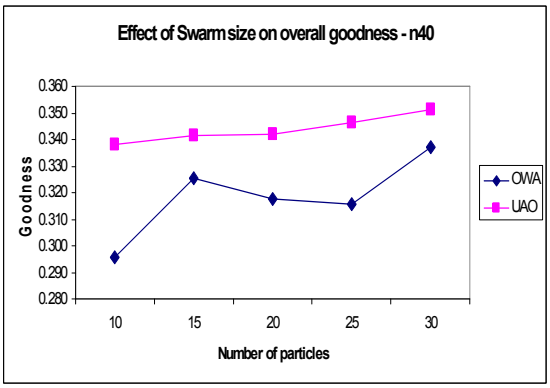
Table 9.8: Results for best and worst average overall goodness and their respective number of particles for UAO. Statistically significant improvement is in italics.

Case	Particles	Max. goodness	Particles	Min. goodness	% improvement
n15	30	0.333 ±0.003	10,15,20,25	0.332 ±0.002	0.42
n25	30	0.339 ±0.008	10	0.330 ±0.009	*2.69*
n33	25	0.339 ±0.005	10	0.332 ±0.006	*2.06*
n40	30	0.351 ±0.012	10	0.338 ±0.005	*3.90*
n50	30	0.336 ±0.003	10	0.333 ±0.005	*0.89*

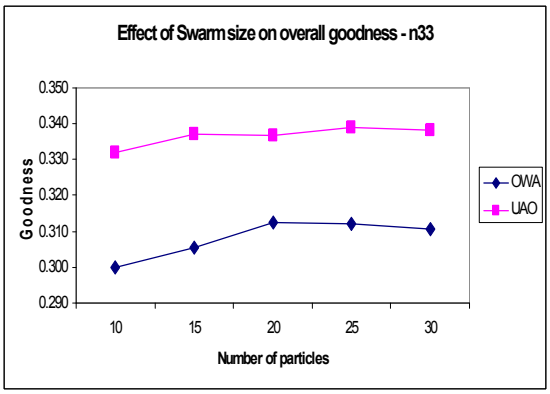
The above discussion and observations suggest that, in general, an increase in



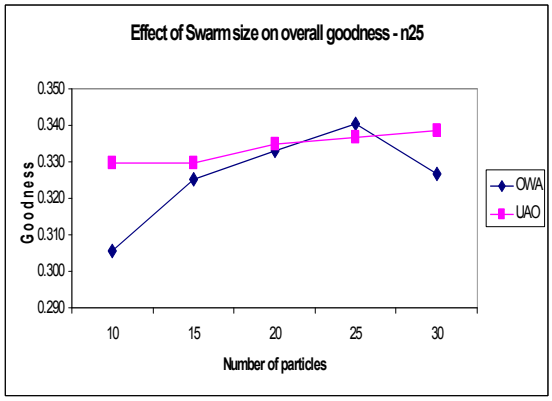
(a)



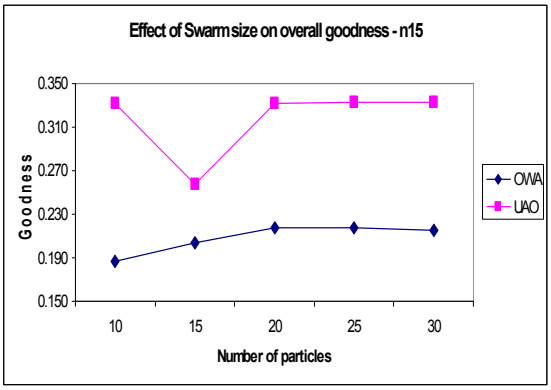
(b)



(c)



(d)



(e)

Figure 9.2: Effect of swarm size on overall goodness for (a) n50 (b) n40 (c) n33 (d) n25 (e) n15

the number of particles increases diversity and reduces the possibility of getting trapped in local minima, thereby resulting in higher quality solutions.

9.2.2 Effect of Acceleration Coefficients

The effect of acceleration coefficients was investigated with different values of the coefficients as given in Table 9.1. Other parameters were kept as follows: number of particles = 20, $w = 0.72$, and $V_{max} = 5$. Tables 9.9 and 9.10 respectively provide the results for OWA and UAO operators with respect to the three sets of acceleration coefficients. The values $c_1 = c_2 = 1.49$ (along with inertia weight = 0.72) were specifically chosen, since they are often used in the literature and they ensure convergence [244].

Table 9.9: Effect of acceleration coefficients on the test cases, for OWA. Good = average overall goodness, Time = Run time (in seconds). % imp shows the improvement achieved by one set of values of c_1 and c_2 over the other set of values. Statistically significant improvement is in italics.

Case	$c_1 = c_2 = 0.5$		$c_1 = c_2 = 1.49$		$c_1 = c_2 = 2.0$		% imp 0.5 vs 1.49	% imp 2.0 vs 1.49	% imp 2.0 vs 0.5
	Good	Time	Good	Time	Good	Time			
n15	0.218 ±0.031	29.8	0.224 ±0.048	29.3	0.218 ±0.043	31.2	-2.95	-2.78	0.0
n25	0.333 ±0.031	118.7	0.333 ±0.027	111.1	0.323 ±0.033	120.5	0.0	-2.91	-2.91
n33	0.313 ±0.053	445.3	0.303 ±0.033	497.9	0.312 ±0.029	546.7	3.14	3.11	-0.03
n40	0.318 ±0.036	1132.9	0.330 ±0.037	1248.5	0.327 ±0.041	1428.8	-4.04	-1.00	2.93
n50	0.263 ±0.039	4291.1	0.262 ±0.052	5285.4	0.275 ±0.043	5444.8	0.58	*4.89*	4.34

It is observed in Table 9.9 that each set of acceleration coefficients produced

Table 9.10: Effect of acceleration coefficients on the test cases, for UAO. Good = average overall goodness, Time = Run time (in seconds). % imp shows the improvement achieved by one set of values of c_1 and c_2 over the other set of values. Statistically significant improvement is in italics.

Case	$c_1 = c_2 = 0.5$		$c_1 = c_2 = 1.49$		$c_1 = c_2 = 2.0$		% imp 0.5 vs 1.49	% imp 2.0 vs 1.49	% imp 2.0 vs 0.5
	Good	Time	Good	Time	Good	Time			
n15	0.332 ±0.001	28.7	0.333 ±0.002	29.7	0.332 ±0.001	32.5	-0.12	-0.12	0.0
n25	0.335 ±0.005	120.3	0.338 ±0.010	119.3	0.337 ±0.009	127.4	-1.03	-0.51	0.52
n33	0.337 ±0.005	455.9	0.338 ±0.006	506.1	0.336 ±0.004	617.7	-0.25	-0.64	-0.39
n40	0.342 ±0.009	1081.8	0.328 ±0.060	1327.2	0.350 ±0.011	1280.5	*3.96*	*6.09*	2.22
n50	0.335 ±0.004	4505.4	0.336 ±0.004	5658.1	0.335 ±0.003	5766.1	-0.13	-0.13	0.0

results of the almost the same quality when compared with the other set. For example, values of $c_1 = c_2 = 0.5$ produced slightly better results than $c_1 = c_2 = 1.49$ for test cases *n50* and *n33*, but the latter set of coefficients performed better than the former for *n40* and *n15*. Similarly, $c_1 = c_2 = 2.0$ produced better results for some cases and worse results for others, when compared with $c_1 = c_2 = 1.49$ and $c_1 = c_2 = 0.5$. However, the t-test showed that, in general, the percentage improvements reported in Table 9.9 were not statistically significant.

With respect to the UAO operator, a trend similar to that of the OWA operator was observed. Table 9.10 shows that the percentage improvements achieved by any set of c_1 and c_2 compared to another set was at most 1% in the majority of cases. An exception from this trend was the case of *n40*, where $c_1 = c_2 = 0.5$ achieved an improvement of 3.96% over $c_1 = c_2 = 1.49$, $c_1 = c_2 = 2.0$ achieved an improvement of 6.09% over $c_1 = c_2 = 1.49$, and $c_1 = c_2 = 2.0$ achieved an improvement of

2.22% over $c_1 = c_2 = 0.5$. A t-test validation showed that all improvements were insignificant, with the exception of *n40* when comparing $c_1 = c_2 = 1.49$ with other sets of c_1 and c_2. In general, the results indicate that the convergence of PSO is independent of the acceleration coefficients with respect to the values used.

9.2.3 Effect of Inertia Weight

Table 9.11: Effect of inertia weight on the test cases, for OWA. Good = average overall goodness, Time = Run time (in seconds). % imp shows the improvement achieved by one value of w over the other value. Statistically significant improvement is in italics.

Case	w = 0.72		w = 0.95		w = 0.4		% imp 0.72 vs 0.95	% imp 0.72 vs 0.4	% imp 0.95 vs 0.4
	Good	Time	Good	Time	Good	Time			
n15	0.218 ±0.031	29.8	0.233 ±0.053	28.9	0.213 ±0.022	28.2	-6.76	2.40	*8.58*
n25	0.333 ±0.031	118.7	0.323 ±0.032	128.5	0.328 ±0.031	125.6	2.88	1.46	-1.46
n33	0.313 ±0.053	445.3	0.291 ±0.026	426.1	0.306 ±0.023	439.9	6.98	2.09	-5.26
n40	0.318 ±0.036	1132.9	0.322 ±0.028	1095.7	0.331 ±0.024	1105.1	-1.49	-4.28	-2.75
n50	0.263 ±0.039	4291.1	0.258 ±0.044	4285.1	0.273 ±0.040	3872.9	1.93	-3.53	-5.57

The effect of the inertia weight, w, is empirically investigated in this section. Tables 9.11 and 9.12 respectively show the results obtained for the fuzzy PSO with the OWA and UAO operators. The effect of w on performance was studied with three values, namely $w = 0.72$, $w = 0.95$, and $w = 0.4$. Other parameters were kept as follows: number of particles = 20, $c_1 = c_2 = 0.5$, and $V_{max} = 5$.

Table 9.11 suggests that there was no clear trend as which value of w produced

Table 9.12: Effect of inertia weight on the test cases, for UAO. Good = average overall goodness, Time = Run time (in seconds). % imp shows the improvement achieved by one value of w over the other value. Statistically significant improvement is in italics.

Case	w = 0.72		w = 0.95		w = 0.4		% imp 0.72 vs 0.95	% imp 0.72 vs 0.4	% imp 0.95 vs 0.4
	Good	Time	Good	Time	Good	Time			
n15	0.332 ±0.001	28.7	0.332 ±0.001	29.1	0.332 ±0.002	27.7	0.0	0.0	0.0
n25	0.335 ±0.005	120.3	0.331 ±0.005	127.2	0.333 ±0.008	125.4	1.03	0.70	-0.33
n33	0.337 ±0.005	455.9	0.337 ±0.005	455.6	0.340 ±0.008	428.1	0.0	-0.81	-0.88
n40	0.342 ±0.009	1081.8	0.344 ±0.011	1103.6	0.345 ±0.011	1025.4	-0.75	-0.91	-0.16
n50	0.335 ±0.004	4505.4	0.335 ±0.003	4528.5	0.335 ±0.004	4345.3	0.0	0.0	0.0

the best results. For example, $w = 0.72$ produced better solutions than $w = 0.95$ for *n25*, *n33*, and *n50*, but for *n15* and *n40*, $w = 0.95$ produced better results than $w = 0.72$. Similarly, comparisons of $w = 0.72$ with $w = 0.4$, and $w = 0.95$ with $w = 0.4$ did not show any clear pattern as which value of w performed better compared to the other. The statistical t-test also showed that none of the improvements, whether achieved by $w = 0.72$, $w = 0.95$, or $w = 0.4$, were significant. As for UAO, the difference between the overall goodness achieved by the three values of inertia weight was generally less than 1% for all test cases. The t-test showed that the improvements were insignificant. These observations suggest that the fuzzy PSO was insensitive to the inertia weight for both the OWA and UAO operators with respect to the three values of w used.

9.2.4 Effect of Velocity Clamping

The effect of velocity clamping was also empirically studied. Tables 9.13 and 9.15 respectively show the results obtained for fuzzy PSO with OWA and UAO operators. The effect was studied with three values of velocity clamping, with one value fixed at $V_{max} = 5$ for all cases, while the other two were variable, proportional to the test case size. These variable values were $\lceil V_{max} = 10\% \rceil$ and $\lceil V_{max} = 20\% \rceil$ of the test case size. The inspiration for taking 10% and 20% size of the test case comes from mutation rates in genetic algorithms. Note that both V_{max} in PSO and mutation rate in GA perturb the solution, and therefore the functions of both parameters is more or less the same. A number of studies [34, 115, 158, 159] have used the mutation rate up to 20% or more. Therefore, the basis of choosing a variable value of V_{max} is this observation. Other PSO parameters were kept as follows: number of particles = 20, $c_1 = c_2 = 0.5$, and inertia weight $w = 0.72$.

Table 9.13: Effect of velocity clamping on the test cases, for OWA. % imp shows the improvement achieved by one value of V_{max} compared to the other value. NA = Not Applicable.

Case	$V_{max} = 5$ Goodness	$V_{max} = 10\%$ Goodness	$V_{max} = 20\%$ Goodness	% imp 5 vs 10%	% imp 5 vs 20%	% imp 10% vs 20%
n15	0.218 ±0.031	0.220 ±0.048	0.212 ±0.038	-1.00	2.46	3.55
n25	0.333 ±0.031	0.328 ±0.036	The value of V_{max} is 5 here	1.54	NA	-1.54
n33	0.313 ±0.053	0.314 ±0.032	0.277 ±0.032	-0.48	*11.36*	*13.36*
n40	0.318 ±0.036	0.334 ±0.038	0.313 ±0.024	-5.30	1.52	6.93
n50	0.263 ±0.039	The value of V_{max} is 5 here	0.266 ±0.038	NA	-1.16	-1.15

Table 9.13 shows that velocity clamping did not significantly improve the so-

Table 9.14: Average algorithm run time (in seconds) for different values of V_{max} given in Table 9.13.

Test Case	Run time		
	$V_{max} = 5$	$V_{max} = 10\%$	$V_{max} = 20\%$
n15	29.8	28.8	28.9
n25	118.7	122.4	Same runtime as for $V_{max} = 5$
n33	445.3	433.0	452.5
n40	1132.9	1075.1	1101.9
n50	4291.1	Same runtime as for $V_{max} = 5$	4386.7

Table 9.15: Effect of velocity clamping on the test cases, for UAO. % imp shows the improvement achieved by one value of V_{max} compared to the other value. NA = Not Applicable.

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

lution. The overall goodness for the test cases varied between 1.0% and 6.93%, with the exception of *n33*. For *n33*, improvements of 11.36% (for comparison of $V_{max} = 5$ with $V_{max} = 20\%$) and 13.36% (for comparison of $V_{max} = 10\%$ with $V_{max} = 20\%$) were obtained. A t-test validation also showed that all improvements less than 13.36% were statistically insignificant. As for UAO, the results in Table 9.15 suggest a trend similar to that of OWA. It is observed in Table 9.15 that velocity clamping had a very slight impact on the quality of overall goodness, with all

Table 9.16: Average algorithm run time (in seconds) for different values of V_{max} given in Table 9.15.

Test Case	Run time		
	$V_{max} = 5$	$V_{max} = 10\%$	$V_{max} = 20\%$
n15	28.7	28.5	28.9
n25	120.3	119.9	Same runtime as for $V_{max} = 5$
n33	455.9	441.7	454.5
n40	1081.8	1098.6	1083.9
n50	4505.4	Same runtime as for $V_{max} = 5$	4867.0

values having less than 1.5% improvements. The t-test confirmed that all the improvements were statistically insignificant. In general, the results in Tables 9.13 and 9.15 suggest that velocity clamping did not have a significant effect on the quality of the overall goodness for the values used for V_{max}.

9.3 Comparison of OWA and UAO

Table 9.17 compares the OWA and UAO operators using linear regression analysis (performed with a confidence level of 95%), with the number of particles as the independent variable and the overall goodness of solution as the dependent variable. The objective of the regression analysis was to study the effect of increasing the number of particles on the overall goodness while using the OWA and UAO operators. The data for the analysis consisted of overall goodness for each particle set in Tables 9.2 to 9.6. Since, for each test case, 30 runs were done for each particle set, the regression coefficients in Table 9.17 was obtained using 150 values (5 values $\times$ 30 runs). For example, for test case *n15*, Table 9.6 shows the overall goodness for

Table 9.17: Comparison of OWA and UAO for FPSO.

Case	Regression coefficients		Ratio = $\frac{UAO}{OWA}$	Comment
	OWA	UAO		
n15	0.309	0.227	0.735	UAO increases goodness slower than OWA at the rate of 0.735
n25	0.238	0.472	1.983	UAO increases goodness almost twice as fast as OWA
n33	0.1	0.329	3.290	UAO increases goodness almost thrice as fast as OWA
n40	0.289	0.392	1.356	UAO increases goodness faster than OWA at the rate of 1.356
n50	0.213	0.254	1.192	UAO increases goodness faster than OWA at the rate of 1.192

OWA and UAO, with five different number of particles (from 10 up to 30). Table 9.17 also provides the regression coefficients for OWA and UAO in columns 2 and 3 respectively. The ratio of UAO and OWA is given in column 4. Based on this ratio, the sensitivity of OWA and UAO was evaluated with respect to the increasing swarm size. An interpretation of the ratio in column 4 is given in column 5.

In Table 9.17, it is observed that UAO was more sensitive than OWA with respect to the increasing number of particles. For almost all cases, UAO increased the overall goodness significantly faster than OWA. This is quite obvious for cases *n25* and *n33* where the rate of UAO is respectively twice and thrice that of OWA. For cases *n40* and *n50*, UAO also had a faster rate than OWA. Only for *n15*, OWA was faster than UAO, as observed in Table 9.17. Thus, the general conclusion is that, as the number of particles are increased, the rate at which UAO increases the overall goodness is significant as compared to the rate of the OWA operator.

9.4 Conclusions

A fuzzy multi-objective particle swarm optimization algorithm for the DLAN topology design problem was proposed and investigated in this chapter. The performance of the algorithm was evaluated with respect to different parameters of the fuzzy PSO algorithm. Results showed that the larger swarm sizes produced better results than medium or small sizes. An investigation of acceleration coefficients suggested that for the three set of values of acceleration coefficients used, there was no significant difference in the quality of final solutions obtained. Results also revealed that the fuzzy PSO was insensitive to the inertia weight, with respect to the three values used. As for velocity clamping, the results suggested that the parameter did not have a significant effect on the quality of the solutions with the three values used. With respect to the performance of OWA and UAO, it was found that, in general, UAO performed better than OWA.

The next chapter presents a comprehensive comparison of the techniques proposed and discussed in this thesis is presented.

Chapter 10

Comparison of Techniques

Chapters 5 to 9 presented a number of iterative algorithms as applied to solve the DLAN topology design problem. This chapter presents an overall comparison of the results obtained for each of the proposed algorithms. As mentioned earlier, the proposed algorithms are divided into two categories: ones which operate on single solutions, such as stochastic evolution, simulated evolution, and simulated annealing, and ones which are population-based swarm intelligence algorithms, namely ant colony optimization and particle swarm optimization. Therefore, the comparisons presented below are also categorized in the same way. The results for these comparisons have been presented and discussed in earlier chapters, but are consolidated here for the sake of clear comparisons.

10.1 Comparison of Single Solution Algorithms

This section compares the performance of the single solution algorithms presented in this thesis. Chapter 5 showed that the variant of variant of StocE with tabu

search characteristics (TFStocE) demonstrated the best performance among different versions of StocE algorithms. For the simulated evolution algorithm, the results in Chapter 6 suggested that the simulated evolution algorithm with tabu search characteristics and dynamic bias (DTFSimE) had the best performance among all proposed variants. As for simulated annealing, the variants were discussed in Chapter 7, and it was found that the simulated annealing algorithm with tabu search characteristics (TEFSA) was the best compared to the other variants. Therefore, this section mutually compares these best algorithms.

Table 10.1: Comparison of TFStocE, DTFSimE, and TEFSA using OWA. % imp denote percentage improvements. Statistically significant improvement is in italics.

Case	TFStocE Goodness	DTFSimE Goodness	TEFSA Goodness	% imp TEFSA vs TFStocE	% imp TEFSA vs DTFSimE	% imp DTFSimE vs TFStocE
n15	0.120 ± 0.053	0.206 ± 0.052	0.389 ± 0.056	*69.15*	*47.04*	*41.75*
n25	0.160 ± 0.035	0.240 ± 0.008	0.500 ± 0.125	*68.00*	*52.00*	*33.33*
n33	0.099 ± 0.045	0.229 ± 0.061	0.429 ± 0.192	*76.92*	*46.62*	*56.77*
n40	0.131 ± 0.053	0.340 ± 0.122	0.489 ± 0.047	*73.21*	*30.47*	*61.47*
n50	0.178 ± 0.044	0.350 ± 0.143	0.329 ± 0.121	*45.90*	-6.38	*49.14*

Table 10.1 compares TFStocE, DTFSimE, and TEFSA with respect to the overall goodness using the OWA operator. The average run time is given in Table 10.2. It is obvious from Table 10.1 that, in general, TEFSA produced the best results among the three schemes, also validated by the t-test. An exception was observed in the case of *n50*, where DTFSimE was able to achieve slightly better results than

Table 10.2: Average run time (in seconds) of algorithms in Table 10.1.

Test Case	Run time		
	TFStocE	DTFSimE	TEFSA
n15	4.6	134.2	89.5
n25	30.1	354.2	314.8
n33	36.2	1080.7	764.7
n40	145.1	2861.0	1499.5
n50	1341.1	7042.8	4295.4

Table 10.3: Comparison of TFStocE, DTFSimE, and TEFSA using UAO. % imp denote percentage improvements. Statistically significant improvement is in italics.

Case	TFStocE Goodness	DTFSimE Goodness	TEFSA Goodness	% imp TEFSA vs TFStocE	% imp TEFSA vs DTFSimE	% imp DTFSimE vs TFStocE
n15	0.245 ± 0.032	0.446 ± 0.061	0.365 ± 0.016	*32.88*	*-22.19*	*45.07*
n25	0.275 ± 0.008	0.301 ± 0.006	0.412 ± 0.064	*33.25*	*26.94*	*8.64*
n33	0.224 ± 0.032	0.303 ± 0.004	0.411 ± 0.072	*45.50*	*26.28*	*26.07*
n40	0.318 ± 0.032	0.297 ± 0.122	0.470 ± 0.079	*32.34*	*36.81*	-7.07
n50	0.256 ± 0.025	0.281 ± 0.000	0.374 ± 0.050	*31.55*	*24.87*	*8.90*

TEFSA. However, this improvement by DTFSimE was not statistically significant. It is also observed that TFStocE was the worst performer among the three schemes, with results much inferior to both DTFSimE and TEFSA.

As for UAO, the trends are very much similar to that of OWA. Observe from Table 10.3 that TEFSA demonstrated the best performance for almost all test cases. The only exception to this was case *n15* where DTFSimE had statistically better performance than TEFSA, as validated by the t-test. Again, TFStocE produced the

Table 10.4: Average run time (in seconds) of algorithms in Table 10.3.

Test Case	Run time		
	TFStocE	DTFSimE	TEFSA
n15	0.7	116.9	88.5
n25	16.0	312.9	322.4
n33	11.3	775.1	757.0
n40	143.1	2526.3	1564.4
n50	54.2	5013.3	3485.6

worst results. However, the extent of degradation in the quality of results produced by TFStocE compared to DTFSimE and TEFSA was not as significant as was observed for OWA. The average run time is given in Table 10.4.

10.2 Comparison of Population Based Algorithms

This section compares the performance of the two population based algorithms, namely, FACO and FPSO, developed in this thesis. The FACO and FPSO algorithms were proposed in Chapters 8 and 9 respectively. Tables 10.5 and 10.7 compares the two algorithms for the OWA and UAO operators respectively. Although the results were discussed in detail in the previous two chapters, these tables present the consolidated best results found in each of the two chapters along with the corresponding parameter setup.

With respect to the OWA operator, Table 10.5 (with average run time given in Table 10.6) suggests that FACO showed statistically better performance than FPSO for the majority of cases, as validated by the t-test. This observation is prominent in cases *n15, n25*, and *n33*. For *n40* and *n50*, FACO had a milder deterioration than FPSO. However, this deterioration was not statistically significant. In general,

Table 10.5: Comparison of FACO and FPSO for OWA. dep = pheromone deposit rate, evap = pheromone evaporation rate, % imp = percentage improvement achieved by FACO. OG = overall goodness. Statistically significant improvement is in italics.

Case	FACO				FPSO					% imp FACO vs FPSO
	ants	dep	evap	OG	par	V_{max}	c_1, c_2	w	OG	
n15	30	0.2	0	0.313 ± 0.032	20	5	0.5	0.95	0.233 ± 0.053	*25.56*
n25	30	0.6	0.2	0.430 ± 0.023	25	5	0.5	0.72	0.340 ± 0.030	*20.93*
n33	30	0.8	0.3	0.362 ± 0.022	20	10%	0.5	0.72	0.314 ± 0.032	*13.26*
n40	25	0.4	0.1	0.333 ± 0.028	30	5	0.5	0.72	0.337 ± 0.026	-1.20
n50	30	0.8	0.3	0.270 ± 0.032	30	5	0.5	0.72	0.287 ± 0.034	-6.30

Table 10.6: Average run time (in seconds) of algorithms in Table 10.5.

Test Case	Run time	
	FACO	FPSO
n15	50.3	28.9
n25	234.0	153.0
n33	553.3	433.0
n40	1521.2	1736.7
n50	5923.3	6328.1

Table 10.7: Comparison of FACO and FPSO for UAO. dep = pheromone deposit rate, evap = pheromone evaporation rate, % imp = percentage improvement achieved by FACO. OG = overall goodness. Statistically significant improvement is in italics.

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

it can be claimed that FACO performed better than FPSO for OWA.

As for UAO, the trends in Table 10.7 (with average run time given in Table 10.8) are somewhat similar to that of OWA. In *n25* and *n33*, FACO showed statistically significant improvement in the quality of overall goodness compared to FPSO. However, for the other three cases, both FACO and FPSO showed equal performance as there was no statistically significant difference in the results. Thus, it can be fairly claimed that FACO also demonstrated better results than FPSO for UAO.

Since the swarm size is a common factor in both FACO and FPSO, it is important to highlight the effect of this factor on the improvement of results. An observation from Tables 10.5 and 10.7 shows the relation of swarm size with the best results: for both FACO and FPSO, it is observed that the best results were obtained for large swarm sizes. For example, for FACO, the best results were obtained when the

Table 10.8: Average run time (in seconds) of algorithms in Table 10.7.

Test Case	Run time	
	FACO	FPSO
n15	31.3	29.7
n25	268.5	187.0
n33	528.1	428.1
n40	1561.9	1659.2
n50	6478.8	7074.9

largest number of ants, i.e. 30, were chosen. There were some instances, such as *n40* in Table 10.5 and *n33* in Table 10.7, where a number of ants equal to 25 produced the best results. As for FPSO, the larger swarm size (25 and 30 particles) produced the best results. There were some instances in both tables where a medium swarm size of 20 produced the best results. In general, it can be said that the swarm size in the two swarm intelligence techniques was directly proportional to the improvement achieved in the quality of results.

10.3 Overall Comparison of OWA and UAO

This section provides an overall comparison of the OWA and UAO operators. This comparison is based on the results presented in Chapters 5 to 9. In Chapter 5, UAO produced much better results than OWA for TFStocE. Results in Chapter 6 revealed that for DTFSimE, UAO performed better than OWA for the number of hops and reliability objectives, and worse than OWA for the delay objective. However, for the cost objective, both UAO and OWA produced results of equal quality. With regard to TEFSA, results in Chapter 7 showed that UAO was better than OWA for the cost objective, and worse than OWA for the delay and number of hops objectives.

As for the reliability objective, both OWA and UAO showed equal performance. As far as FACO and FPSO are concerned, results in Chapters 8 and 9 suggested that UAO performed better than OWA for both algorithms. From the above discussion, it can be fairly claimed that UAO is preferred to OWA.

10.4 Overall Best Algorithm

In Section 10.1, it was found that in the category of single solution algorithms, TEFSA produced the best results among the three algorithms. As for the category of population based algorithms, the results in Section 10.2 suggested that FACO produced better results than FPSO. However, it will be interesting to compare the two best algorithms from each category. Tables 10.9 and 10.10 provide a comparison of TEFSA and FACO by using the OWA and UAO operators respectively. It is observed from both tables that TEFSA achieved statistically significant improvement over FACO for all test cases, and for both OWA and UAO.

Table 10.9: Comparison of FACO and TEFSA for OWA. dep = pheromone deposit rate, evap = pheromone evaporation rate, Time = run time (in seconds), % imp = percentage improvement achieved by TEFSA. Statistically significant improvement is in italics.

Case	FACO					TEFSA		% imp
	ants	dep	evap	Goodness	Time	Goodness	Time	
n15	30	0.2	0	0.313 ±0.032	50.3	0.389 ±0.056	89.5	*19.54*
n25	30	0.6	0.2	0.430 ±0.023	234.0	0.500 ±0.125	314.8	*14.00*
n33	30	0.8	0.3	0.362 ±0.022	553.3	0.429 ±0.192	764.7	*15.62*
n40	25	0.4	0.1	0.333 ±0.028	1521.2	0.489 ±0.047	1499.5	*31.90*
n50	30	0.8	0.3	0.270 ±0.032	5923.3	0.329 ±0.121	4295.4	*17.93*

Table 10.10: Comparison of FACO and TEFSA for UAO. dep = pheromone deposit rate, evap = pheromone evaporation rate, Time = run time (in seconds), % imp = percentage improvement achieved by TEFSA. Statistically significant improvement is in italics.

Case	FACO					TEFSA		% imp
	ants	dep	evap	Goodness	Time	Goodness	Time	
n15	30	0.8	0.3	0.334 ±0.002	31.3	0.365 ±0.016	88.5	*8.49*
n25	30	0.4	0.1	0.363 ±0.008	268.5	0.412 ±0.064	322.4	*11.89*
n33	25	0.8	0.3	0.349 ±0.006	528.1	0.411 ±0.072	757.0	*15.09*
n40	30	0.6	0.2	0.352 ±0.006	1561.9	0.470 ±0.079	1564.4	*25.11*
n50	30	0.6	0.2	0.336 ±0.004	6478.8	0.374 ±0.050	3485.6	*10.16*

10.5 Conclusion

An overall comparison of the proposed techniques was presented in this chapter. Since simulated annealing, simulated evolution, and stochastic evolution are algorithms which operate on a single solution, their best results were mutually compared. The comparison revealed that among TFStocE, DTFSimE, and TEFSA, it was TEFSA that generally produced the best results. On the other hand, ant colony optimization and particle swarm optimization are two swarm intelligence techniques that maintain and evolve a collection of candidate solutions. A comparison of the two swarm-based approaches showed that FACO had a better performance than FPSO. An overall comparison showed that TEFSA produced the best results among all algorithms proposed in this thesis. Moreover, an overall comparison of OWA and UAO showed that UAO performed better than OWA.

The next chapter provides a brief summary of the thesis and some directions for future research.

Chapter 11

Conclusion

This thesis addressed the problem of the topology design of distributed local area networks, modelled as a multi-objective optimization problem. The first main objective of the thesis was the design and analysis of some iterative heuristics and swarm intelligence algorithms to address the DLAN topology design problem. This objective was accomplished by engineering a number of algorithms, such as simulated evolution, stochastic evolution, simulated annealing, ant colony optimization, and particle swarm optimization for the DLAN topology design problem. The second main objective was to address the multi-objective nature of the problem, and was accomplished by using fuzzy logic to aggregate individual objectives into a multi-objective aggregation function. Hybridization of single-solution algorithms was also investigated and new hybrid algorithms for the DLAN topology design problem were proposed and evaluated.

The following sections briefly highlight the key findings and contributions of this thesis, followed by a short discussion on directions for future research.

11.1 Summary

Chapter 3 provided details on the formulation of a multi-objective topology design problem for distributed local area networks. All necessary problem-specific information, including assumptions, objectives, and constraints, were discussed in the chapter. The chapter also discussed the integration of the multiple design objectives into a single objective function using fuzzy logic.

Chapter 4 proposed and discussed a new fuzzy aggregation function, namely the unified and-or (UAO) operator. The proposed UAO operator was theoretically and empirically compared with the well-known ordered weighted average (OWA) operator. The UAO operator exhibited mathematical properties similar to that of the OWA operator, and empirically performed better than OWA for the test instances. The chapter also discussed a structured decision-making approach based on fuzzy logic, with specific application to the problem of topology design of distributed local area networks.

Chapter 5 proposed a fuzzy multi-objective technique based on the stochastic evolution algorithm, termed as 'FStocE'. A variant of the proposed stochastic evolution algorithm, 'TFStocE', was also proposed. This variant introduced tabu search characteristics in the FStocE algorithm. The two variants were mutually compared empirically, using the OWA and UAO operators. It was found that, in general, TFStocE produced better results than FStocE for both the OWA and UAO operators. Moreover, an investigation was also done on a dynamic value of R, which is an important parameter of the standard stochastic evolution algorithm. The results suggested that the proposed approach for computing a dynamic R produced inferior

solutions compared to the TFStocE algorithm, for both the OWA and UAO operators. As far as the effectiveness of the OWA and UAO operators are concerned, the investigation found that UAO performed much better than OWA in optimizing each of the four design objectives.

Chapter 6 proposed and investigated a fuzzy multi-objective algorithm based on a simulated evolution algorithm, namely FSimE. A variant of FSimE, known as 'TFSimE', was also proposed. TFSimE incorporated tabu search characteristics in the *allocation* phase of the FSimE algorithm. The comparison suggested that TFSimE generally produced better results than FSimE. This improvement was observed in both cases when OWA and UAO operators were used for aggregation of single objectives. Another issue investigated in the chapter was the usage of dynamic *bias* value. Since bias is the only parameter in the simulated evolution algorithm, its proper value has an impact on the quality of the solution. Results suggested that the proposed approach for TFSimE based on dynamic bias value, denoted as DTFSimE, produced better results than TFSimE for all test cases with respect to OWA and the majority of test cases for UAO. As far as the relative performance of the OWA and UAO operators was concerned, it was found that UAO performed much better than OWA for the *number of hops* and *reliability* design objectives, while UAO had an inferior performance for the *cost* and *delay* objectives.

Chapter 7 investigated the effectiveness of a fuzzy simulated annealing algorithm for the DLAN topology design problem. A fuzzy simulated annealing algorithm, termed 'FSA', was proposed. Two variants of FSA were also developed. The first variant introduced tabu search characteristics into FSA, and was named TFSA. The second variant, namely TEFSA, introduced simulated evolution characteristics into

TFSA. A comparison of the three variants of the simulated annealing algorithms revealed that, generally, TEFSA produced better results than TFSA and FSA. This trend was observed for both the OWA and UAO operators. Moreover, mutual comparison of OWA and UAO with respect to the four design objectives suggested that both operators had more or less similar results. Another issue discussed in the chapter was the proposal of a dynamic value of an important parameter of the simulated annealing algorithm, namely the Markov chain, M. Although the proposed computation of dynamic M produced lower quality results than FSA, degradation in quality was not very great.

Chapter 8 proposed and discussed a fuzzy multi-objective ant colony optimization algorithm. Since heuristic value is an important factor of the ant colony algorithm, a fuzzy heuristic value was proposed. Furthermore, since pheromone deposit and evaporation, and the number of ants, are important parameters that play a key role in the algorithm's search direction, empirical analysis was done to study the effect of these parameters. As far as pheromone deposit and evaporation was concerned, better results were obtained when the difference in pheromone deposit and evaporation rates was high. This was a general trend for both OWA and UAO. As for the number of ants, relatively better results were obtained when the number was high. As for the mutual comparison of OWA and UAO, results suggested that UAO produced better results in the context of its usage in the fuzzy ant colony optimization algorithm.

Chapter 9 presented a fuzzy multi-objective particle swarm optimization algorithm for the DLAN topology design problem. A preliminary analysis of the effect of the number of PSO parameters was provided. With respect to swarm sizes, it

was found that larger sizes produced the best results compared to medium or small sizes. With respect to acceleration coefficients, results suggested that having different values of acceleration coefficients had no significant effect on the quality of final solutions obtained, with respect to the three values tested. A similar observation was made for inertia weight, where results revealed that the fuzzy PSO was insensitive to the inertia weight to a considerable extent, for the three values of the inertia weight used. As for velocity clamping, results suggest that the parameter also did not have a very significant effect on the quality of the solution with respect to the tested values. With respect to the performance of OWA and UAO, it was found that, in general, UAO performed better than OWA.

Chapter 10 provided an overall comparison of the techniques proposed in this thesis. Since simulated annealing, simulated evolution, and stochastic evolution are algorithms which operate on a single solution, their best results were mutually compared. On the other hand, ant colony optimization and particle swarm optimization are two swarm intelligence techniques, which were compared with one another. The comparison revealed that among TFStocE, DTFSimE, and TEFSA, the general trend is that TEFSA produced the best results. As for FACO and FPSO, the comparison suggested that FACO had a better performance than FPSO. Also, an overall comparison showed that TEFSA produced the best results among all algorithms proposed in this thesis. Moreover, an overall comparison of OWA and UAO showed that UAO had better performance than OWA.

11.2 Future Research

Some directions for future research are summarized below.

Efficient Approaches for Dynamic Parameters

Chapters 5, 6, and 7 discussed some approaches to dynamically adjust parameter values. However, most of these approaches did not prove to be efficient in producing high-quality solutions. More research is needed to devise highly efficient ways to deal with dynamic assignment of parameter values. Furthermore, the parameters in ACO and PSO are statically assigned by the user. Some research is also needed to develop mechanisms to dynamically adjust their parameter values.

Hybridization of tabu search with ACO and PSO

Results in this thesis suggested that introducing tabu search characteristics into different algorithms resulted in better solutions. However, the hybridization was done only on algorithms which operate on single solutions, such as simulated annealing, simulated evolution, and stochastic evolution. The promising results encourage research on incorporating tabu search characteristics or features of simulated evolution into swarm intelligence algorithms.

Extension of Fuzzy PSO to the *lbest* Model

The fuzzy PSO introduced in Chapter 9 made use of the *gbest* PSO model. Results for the proposed fuzzy PSO suggested that the algorithm was not very effective as compared to ACO. Effectiveness of fuzzy PSO can possibly be enhanced by extending the algorithm for the *lbest* model, since previous research has suggested that the *lbest* model may be more effective due to its better ability to escape local minima. Therefore, a recommendation is to exploit the *lbest* model for fuzzy PSO.

More In-depth Study of the Effects of the PSO parameters

Chapter 9 provided a preliminary analysis of the FPSO algorithm with at most three values for each parameter. More in-depth study is required to asses the effects of the PSO paramterers.

Application of Other Techniques to DLAN Topology Design Problem

Other techniques such as genetic algorithms, differential evolution, and estimation of distribution methods have been applied to a number of optimization problems. A recommendation is that these techniques be engineered for the DLAN topology design problem and that a comparative study be done with the techniques proposed in this thesis.

Other Aggregation Techniques

The multi-objective aspects of the DLAN topology design problem presented in this paper were addressed by fuzzy logic based aggregation functions such as UAO and OWA. However, other techniques presented in Chapter 2 should also be exploited.

Application of the UAO Operator to other Multi-objective Problems

The study of the UAO operator showed better performance as compared to the OWA operator for the DLAN topology design problem. Therefore, application of UAO to other multi-objective optimization problems should also be studied.