2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th Intl. Conference on Data Science and Systems

Optimizing Infrastructure Placement in Wireless Mesh Networks using NSGA-II

Liqaa F. Nawaf, Stuart M. Allen, Omer Rana
School of Computer Science & Informatics,
Cardiff Metropolitan and Cardiff University, UK
LLLNawaf@cardiffmet.ac.uk, {AllenSM,RanaOF}@cardiff.ac.uk

***Abstract*—Wireless Mesh Networks (WMNs) provide a flexible and low-cost technology to efficiently deliver broadband services to communities. In a WMN, a mesh router is deployed at each house, which acts both as a local access point and a relay to other nearby houses. Since mesh routers typically consist of off-the-shelf equipment, the major cost of the network is in the placement and management of Internet Transit Access Points (ITAP) which act as the connection to the internet. In designing a WMN, the aim is to minimize the number of ITAPs required whilst maximizing the traffic that could be served to each house. A multi-objective optimization algorithm is investigated to solve the WMN infrastructure placement problem, using crossover and mutation operators. A simulation based analysis is used to demonstrate the benefit of the proposed approach.**

***Index Terms*—Wireless mesh network; multi-objective optimization algorithm; optimization; neighbourhood move; NSGA-II;**

I. Introduction

Wireless Mesh Networks (WMNs) are a promising approach to providing ubiquitous broadband internet access, due to their potential to support high data rates with small infrastructure costs. In an infrastructure WMN, a limited number of Internet Transit Access Points (ITAPs) serve as gateways or bridges to the Internet. Antennae and low-cost mesh routers hosted within residential communities have two functions: i) routing traffic in/out of residential properties (houses) to mesh clients; and ii) acting as relay links in a multi-hop wireless backbone to route traffic throughout the (residential) neighbourhood, communicating with the Internet via ITAPs. Such a multi-hop structure decreases the number of ITAPs needed, leading to reduced operational costs and more effective use of the available decentralized communications infrastructure. An ITAP will share its Internet connection wirelessly with all the houses in its neighbourhood. Each house then shares the connection wirelessly with other houses nearby. This forms networks of varying sizes to serve urban communities within a city or rural area. In wireless neighbourhood networks, a set of houses and a set of ITAPs are designed and deployed. Fixed capacities are associated with each house and ITAP and with all the connecting edges in the network. Demand from houses needs to meet two specific metrics: throughput and fairness, whilst minimising the number of ITAPs. In order to achieve a good trade-off between throughput and fairness a multi-objective optimization (MOO) algorithm is considered. The paper is organized as follow: section II provides literature review of related approaches. Section III defines the network model and section IV briefly highlights the integer linear program formulation of our problem. In section V we discuss techniques for multi-objective optimization and for defining a series of crossover and mutation operators. Section VI reports the experimental results of using the Non-Dominated Sorting Genetic Algorithm (NSGA-II), with Section VII providing conclusions.

II. Related Work

Evolutionary Algorithms (EAs) support multi-objective optimisation problems by maintaining a population of several solutions throughout the optimization process. Some of these populations are good in cost, while others are good in throughput. Combining the population may have some opportunity to introduce diversity. EAs have the potential of finding multiple Pareto-optimal solutions in a single simulation run. In [1] the benefits of Multi-objective Evolutionary Algorithms (MOEAs) are considered. Population-based approaches such as Genetic Algorithms (GAs) can be extended to solve MOO problems referred to as called MOEAs.

Camelo et al. [2] proposed a method for solving routing problems by considering Quality of Service (QoS) in WMNs, using the multi-objective approach relying on EAs. In this study, the NSGA-II was used for finding different alternative routes that guaranteed the QoS requirements in both the Voice over Internet Protocol (VoIP) and file transfer. The results demonstrate the optimal route found by NSGA-II. The NSGA was one of the first such EAs, proposed by Deb et al. in [3]. Over time an improved version of NSGA was introduced, called NSGA-II, enhancing the convergence and the spread of the solutions. Researchers in [2] and [4] suggest and support the use of NSGA-II and show that it has been able to come closer than other EAs to the true Pareto front. For a highly effective way of finding a set of effective solutions, multi-objective approaches may be considered for solving the WMN optimization problem where the overall number of ITAPs is minimized laterally with high throughput and maximum fairness. We also make use of NSGA-II, with a key focus on using crossover and mutation operators suitable for this problem.

978-1-5386-6614-2/18/$31.00 ©2018 IEEE
DOI 10.1109/HPCC/SmartCity/DSS.2018.00271

III. Network Model

The ideal link model proposed in [5] is used with the aim to minimise the number of ITAPs required (without any environment interference), while maximising the bandwidth available to users, and identifying a solution with the *fairest* possible allocation (i.e. all users/ houses benefit from the approach, with no disproportiate benefit for a small group of users). Following [6], a network is formed by a set of houses $H = \{h_1, ...h_M\}$, along with a set *(I)* of *N* locations at which ITAPs can be installed. Each node has a location (x, y). Each house *h* has a traffic demand, w_h, and it may be said that a house is served if all traffic at this location can be successfully transmitted to an active ITAP (possibly through a sequence of hops). It is assumed that the traffic from each house can be subdivided and routed along multiple paths simultaneously; hence a maximum flow algorithm is run to compute the satisfied demand under an ideal link model. A directed graph *G* is constructed with $(H \cup I)$ nodes, with edges joining each pair of nodes that are within wireless range based on distance. The capacity of each edge $e \in E\ (G)$, Cap_e, is the data rate that can be sustained on this link, and each node has a capacity Cap_h which denotes the ability of a house to process and forward data. Each ITAP also has a capacity limit, based on its connection to the Internet and its processing speed, denoted Cap_i.

IV. Integer Linear Programming Formulation

Following [6] the model is formally described and its variables and constraints defined. For each edge *e* and house *h*, a variable $x_{e,h}$ is defined to indicate the flow which originated from *h* to the ITAPs that are routed through *e*. For each ITAP *i*, a variable y_i indicates the number of ITAPs installed at location *i*. As shown in Formulation 1, constraint 1 ensures flow conservation, namely, for every house except the house where the flow originated, the total amount of flow entering the house is equal to the total amount of flow exiting it. The constraint in Equation 2 indicates that a house does not receive the flow sent by itself. Constraints (3 - 5) of the integer program capture the capacity constraints on the edges, houses and ITAPs. Equation 6 says that no house is allowed to send any traffic to an ITAP unless there is sufficient capacity from the ITAPs installed there. The inequality 7 constrains flows to be positive, and follows from the ITAP capacity constraint and the assumption that is an integer in Equation 8. Constraint 9 specifies that the flow from a house must be equal to or less than demand scaled by the allocation for the house. Constraint 10 indicates the proportion of demand that each house is allocated, specifically between [0, 1]. Making the bandwidth b_h means that the bandwidth allocated to each house will be at most $b_h w_h$. Formulation 1:

Minimise $f_l(x)$; Maximise $f_D(x), f_U(x)$ – subject to:

$$\sum\nolimits_{e=(v,h')} x_{e,h} = \sum\nolimits_{e=(h',v)} x_{e,h} \qquad \forall h, h' \in H, h' \neq h \quad (1)$$

$$\sum\nolimits_{e=(v,h)} x_{e,h} = 0 \qquad \forall h \in H \quad (2)$$

$$\sum\nolimits_{h} x_{e,h} \leq Cap_e \qquad \forall e \in E(G) \quad (3)$$

$$\sum\nolimits_{h',e=(v,h)} x_{e,h'} \leq Cap_h \qquad \forall h \in H \quad (4)$$

$$\sum\nolimits_{h',e=(v,i)} x_{e,h'} \leq Cap_i y_i \qquad \forall i \in I \quad (5)$$

$$\sum\nolimits_{e=(v,i)} x_{e,h} \leq w_h y_i \qquad \forall i \in I, h \in H \quad (6)$$

$$x_{e,h} \geq 0 \qquad \forall e \in E(G), h \in H \quad (7)$$

$$y_i \in \{0, 1, 2, ...\} \qquad \forall i \in I \quad (8)$$

$$\sum\nolimits_{e=(h,v)} x_{e,h} \leq b_h w_h \qquad \forall h \in H \quad (9)$$

$$b_h \in [0, 1] \qquad \forall h \in H \quad (10)$$

V. Techniques for Multi-Objective Optimization

We define a chromosome structure (within a GA) for a data set with M houses and N potential ITAP locations. A vector of length $M + N$ is considered, where the first M elements specify the bandwidth allocated to each house and the remaining N elements specify the number of ITAPs to be installed at each location. From Figure 1, a chromosome p:

$p[i]$ is the bandwidth allocation to house i, for $1 \leq i \leq M$

$p[M + i]$ is the number of ITAPs installed at location i, for $1 \leq i \leq N$

For single objective problems as in [7], a fitness value can be calculated for each chromosome, whereas for multi-objective problems the fitness value is replaced by a vector of fitness values for each objective. For the problem of optimizing WMN infrastructure placement, objectives f_D, f_U and f_I are defined in Equations 11, 12 and 13. Equation 14 indicates the minimum of all demand.

$$f_D(x) = \frac{\left(\left(\sum_{h=1}^{M} w_h\right) - \left(\sum_{h=1}^{M} \sum_{e=h,v} x_{e,h}\right)\right)}{\sum_{h=1}^{M} w_h} \quad (11)$$

$$f_U(x) = \frac{[A - \min_{1 \leq h \leq M}(\sum_{e=h,v} x_{e,h})]}{A} \quad (12)$$

$$f_l(x) = \frac{\sum_{i=1}^{N} y_i \times Cap_i}{\sum_{h=1}^{M} w_h} \quad (13)$$

$$A = \min_{1 \leq h \leq M} w_h \quad (14)$$

A GA maintains a population of solutions which are updated through a process of reproduction over a number of generations. The overall aim is that the structures in chromosomes that correspond to "good" solutions should survive and propagate into the next generation. New individuals/ solutions (termed *children*) are created through the application of *crossover* and mutation operators to a pair of *parent* chromosomes. Crossover operators blend the genetic information between a pair of parent chromosomes to explore the search space, whereas mutation operators are used to maintain sufficient diversity in the population. Crossover allows the basic genetic material of parents to pass to their children, who then form the next generation. A number of crossover operators have been proposed for use in GAs, but in almost all of these,

pairs of each gene from the parent chromosomes are combined to pass the corresponding gene on to the child. In NSGA-II, *elitism* is used in building the next generation. The elitism operator combines the old population with the newly created population and chooses to keep the better solutions from the combined population. Elitism can speed up the performance of the GA significantly and this can also help to prevent the loss of good solutions once they are found.

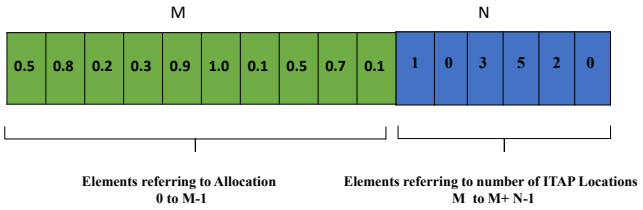

Fig. 1. Single Chromosome of Allocation and Placement

A. Crossover Operator

Crossover operators create new offspring by "mating" two selected parents with the aim of maintaining beneficial structures in the children. Selecting and implementing a crossover operator depends on the chromosome representation and also on the optimization problem. Initially two types of crossover operator are applied to WMN:

Arithmetic crossovers are commonly applied in real-coded GAs; they work by taking the weighted average of the two parents. In [8] arithmetic crossover generates a high number of individuals in the search space and creates a greater variety of individuals by increasing the "genetic diversity" of the population while still maintaining adequate coverage of the ranges near and between the parents. Arithmetic crossover uses the arithmetic mean to produce individuals in the next generation, as shown in Algorithm 1. Previous studies have shown that arithmetic crossover can enhance the rate of convergence [8]. $U(s)$ denotes a uniformly random value from the set s.

Uniform crossover [9] for each gene makes a random, binary decision on which parent to select, based on a specific mixing ratio. For example, with a mixing ratio of 0.5 the child has an equal probability of receiving a given gene from either parent, whereas for a ratio of 0.75, selection is biased towards the first parent. In this paper, the mixing ratio is fixed at 0.5. This is described formally in Algorithm 2.

B. Mutation Operator

Mutation is a genetic operator used to introduce diversity into the population as meta-heuristic algorithms are liable to get stuck in a local optimum. The best mutation rate is often difficult to determine, since a small value may not introduce sufficient diversity, while a large value leads to many offspring, leading to a random walk in the search space. Two mutation operators: *Gaussian* and *Uniform*, are tested here. These types of mutation operator can be used only for integer and real valued genes.

Algorithm 1: Arithmetic Crossover (p, q, M, N)

p, q: parent chromosomes //(Select p and q randomly from population)
M, N: number of houses, potential ITAP locations
$\sigma = U([0, 1])$
c is an empty chromosome of length $M + N$ *#initialize child to Empty list*
Loop over Houses, build allocations for child from parents
for $i \in \{1, ..., M\}$ **do**
  $c[i] = \sigma.p[i] + (1 - \sigma).q[i]$
end
Loop over Placement, build placement for child from parents
for $i \in \{M + 1, ..., M + N\}$ **do**
  $c[i] = \lfloor \sigma.p[i] + (1 - \sigma).q[i] \rfloor$
end
Return c

Algorithm 2: Uniform Crossover (p, q, M, N)

p, q: parent chromosomes //(Select p and q randomly from population)
M, N: number of houses, potential ITAP locations
c is an empty chromosome of length $M + N$ // *initialize child to Empty list*
for $i \in \{1, ..., M + N\}$ **do**
  if $U([0, 1]) > 0.5$ **then**
    $c[i] = p[i]$
  end
  else
    $c[i] = q[i]$
  end
end
Return c

Gaussian Mutation operator adds a Gaussian distributed random value to the chosen gene of the allocation and placement chromosomes, as shown in Algorithm 3. The aim of Gaussian mutation is to avoid being trapped in the local minimum by having more chance to pick genes (a small number of solutions is generated) close to the current solution rather than anywhere else. Mutating the gene preserves more of the current solution [10], and applies diversity on Gaussian principles on a smaller scale. Since allocation values must fall within the range [0, 1], if the mutated value falls outside these limits, it is "clipped" to the maximum/minimum allowed. For the genes corresponding to ITAP placement, clipping is necessary only for the lower bound of 0. However, as Gaussian mutation leads to non-integral values, the resulting values are rounded downwards. The effect of Gaussian mutation is controlled by the standard deviation. The notation of λ_P and λ_A represents the gene mutation probabilities of placement and allocation. In Gaussian mutation, the ways to mutate genes for placement and allocation are specified below;

- Each gene corresponding to ITAP placement is modified with probability λ_P/N.
- Each gene corresponding to bandwidth allocation is modified with probability λ_A/M.

where P denotes the population, P_m represents the mutation rate, σ_P represent the standard deviation for placement and σ_A represents the standard deviation for the allocation.

Algorithm 3: Gaussian Mutate (P, P_m, σ_P, σ_A, λ_P, λ_A, M, N)

p, q: parent chromosomes //(Select p and q randomly from population)
M, N: number of houses, potential ITAP locations
c is an empty chromosome of length $M + N$ // *initialize child to Empty list*
Loop over Houses;
for $i \in \{1, ..., M\}$ **do**
  Checking Mutation rate P_m;
  if $U([0, 1]) < \lambda_A/M$ **then**
    Gaussian mutation & clipping ;
    $c[i] = max(min(c[i] + N(0, \sigma_A), 1.0), 0)$
  end
end
Loop over placement;
for $i \in \{M + 1, ..., M + N\}$ **do**
  Checking Mutation rate P_m;
  if $U([0, 1]) < \lambda_p/N$ **then**
    Gaussian mutation & clipping ;
    $c[i] = max(c[i] + \lfloor N(0, \sigma_P) \rfloor, 0)$
  end
end

Uniform Mutation considers each gene in turn and makes a decision whether to modify each gene separately. Normally, this operation replaces the value of the gene with a value selected uniformly randomly between some upper and lower bounds, as described in Algorithm 4. However, since there is no upper bound on the number of ITAPs that can be installed at a location, separate gene mutation rates and processes for placement and allocation are applied. For ITAP placement a small perturbation is made to the value, rather than selecting an entirely new value.

- Each gene corresponding to ITAP placement is modified with probability λ_P/N, i.e. randomly select to either add 1 to or subtract 1 from the original gene.
- Each gene corresponding to bandwidth allocation is modified with a given probability, i.e. select a random value of delta and then replace the original gene value for each gene of the allocation chromosome.

Each of these two forms of mutation are selected randomly. First, with the probability λ_A/M, the bandwidth allocated to a house may be set at a random value between 0 and 1. If this change is not made, with probability λ_A/M, the bandwidth allocated to the house is set to 0 (i.e. choosing not to serve them at all).

Algorithm 4: Uniform Mutate (P, P_m, λ_P, λ_A M, N)

p, q: parent chromosomes //(Select p and q randomly from population)
M, N: number of houses, potential ITAP locations
c is an empty chromosome of length $M + N$ // *initialize child to Empty list*
Loop over Houses;
for $i \in \{1, ..., M\}$ **do**
  Checking Mutation rate P_m
  if $U([0, 1]) < \lambda_A/M$ **then**
    $c[i] = 0$
  else if $U([0, 1]) < \lambda_A/M$ **then**
    Mutate Gene
    $c[i] = U([0, 1])$
end
Loop over placement;
for $i \in \{M + 1, ..., M + N\}$ **do**
  Checking Mutation rate P_m
  if $U([0, 1]) < \lambda_p/N$ **then**
    Mutate Gene & clipping
    $c[i] = max(c[i] + U(\{-1, 1\}), 0)$
  end
end

TABLE I
BENCHMARK DATA SETS

Data Set	Houses	ITAPs	ITAP Locations	Grid Area
DS1	100	10	10	100×100
DS1A	100	10	10	100×100
DS7	1000	50	100	500×500
DS7A	1000	50	100	500×500

VI. Experimental Results

NSGA-II algorithm was run for the parameters below to test the performance of different combinations of crossover and mutation operators for the data sets described in Table I. Each experiments was run 5 times on each test case with five different random seeds. The results are presented for the mean values of the 5 runs. The experimental results are discussed in the following sections.

A. Parameter Settings

The arithmetic and uniform crossover with uniform and Gaussian mutation were implemented and tested for different combinations of population size and generations with a population ranging from 16 to 200 and a generational range from 125 to 500. A population size of 32 and max generation of 500 are chosen, with a mutation rate (P_m) of 0.1 and gene mutation probabilities of $\lambda_P = \lambda_A = 1$. The data sets in Table I were applied for the wide range of instances for the experiments in Table II. DS1 and DS7 were regenerated with different random seeds for sets of ITAP and house locations to generate new data sets with the same density, i.e. DS1A and DS7A, with the properties as shown in Table I.

Some initial experiments were performed to determine a population size and number of generations that would be ap-

TABLE II
EXPERIMENTS ON DATA SETS

Experiments	Data sets	Wireless Range Connectivity	Wireless Link Capacity	ITAP Capacity	Generation
E6.1	**DS1**	25	5	10	500
E6.2	**DS1A**	25	5	10	500
E6.3	**DS7**	35	15	20	500
E6.4	**DS7A**	35	15	20	500
E6.5	**DS7A**	35	15	20	1000
E6.6	**DS7**	35	54	20	500

TABLE III
EXPERIMENT PARAMETERS

Parameter	P_m	σ_P	σ_A	λ_P	λ_A	**House demand**
Value	0.1	1	1/6	1	1	1

propriate – a smaller population leads to quicker convergence but the algorithm is more likely to get trapped in local optima; conversely a large population affects the ability of the GA to explore the whole search space equally. Based on sensitivity analysis of the results the population size was reduced and run for a longer time, for more generations. To see the progress of the algorithm, a population of 32 individuals and 500 generations were applied as a reasonable balance between quality and runtime in the final solution.

B. Crossover Operator

The first experiments aimed to investigate the effectiveness of the crossover operator. Both arithmetic and uniform crossover were applied with Gaussian mutation operators using the parameters in Table III.

To compare the final populations produced by each crossover, we measure the relative spread of solutions between two sets of solutions – referred to as "set coverage" (to measure any improvement in solutions). Five runs with different random seeds were generated for each crossover type to give sets of populations A and B and both sets were compared pairwise. The average of set coverage indicating uniform crossover outperformed the arithmetic crossover in DS1 and DS1A of E6.1 and E6.2, and the arithmetic crossover outperformed the uniform crossover in DS7 and DS7A of E6.3, E6.4 and E6.6. The assumption is that uniform crossover does better in small data sets, which may be easy problems, whereas arithmetic does better in larger data sets because the problem then is harder.

Effectively, with simulating smaller problems it is easy to achieve improvement and easy to explore the search space but the visible effect is small. Given the suspicion that a small data set denotes an easy problem, it was proposed to investigate harder problems. To confirm and draw attention to the performance of arithmetic crossovers in comparison to uniform crossovers in DS7 and DS7A, the NSGA-II algorithm was run once with 1000 generations for DS7A, to compare uniform and arithmetic crossovers. The longer the experiment ran, the greater the difference found between the two types of crossover. As before, set coverage was compared after every 100 generations. The uniform crossover initially outperforms arithmetic, but arithmetic does better with the remaining generations. This indicates the good progress in larger problems of arithmetic crossover compared to uniform crossover and confirms that arithmetic crossover is more effective with larger data sets.

In [8] the evaluation results show that algorithms which use the arithmetic crossover consistently outperform those using the uniform crossover; the arithmetic crossover is consistently able to reach the neighbourhood of global minima with competitive speeds of convergence. It is clear from the above evaluation result that, while NSGA-II with arithmetic crossover had the highest average set coverage in all test case of DS7 and DS7A, uniform crossover performed well in small data sets such as DS1and DS1A. Hence, further investigation was suggested into the use of uniform crossover in small data sets and arithmetic crossover in larger data sets. From the above experiments where arithmetic and uniform crossover were tested, the focus was determined as uniform crossover in the house allocation, to improve allocation and serve more demands, since arithmetic crossover would pull everyone towards the middle, because arithmetic crossover cannot give anyone either a full allocation or no allocation at all. In other words, it would be hard to make progress in this direction. For the ITAP placement, arithmetic crossover was focused on improving the attainment of this objective. From the techniques of uniform and arithmetic crossover. It is clear that uniform crossover has a chance of selecting a bad solution or an empty gene, while arithmetic crossover tends to move away from selecting a bad solution. This is an advantage, but its downside is that it also tends to move away from good solutions. The arithmetic crossover takes the average of the parents so as to strike a balance between their strengths which smooths it out; this is a better method than taking one another, as in uniform crossover.

The evaluation results indicate that uniform crossover worked best on small data sets and arithmetic crossover had the best average set coverage for larger data sets; therefore arithmetic crossover and uniform crossover were subjected to further investigation.

C. Mutation Operator Experimentation

To compare the performance of the proposed mutation operators, experiments were performed on the data sets of Table I to explore the efficiency of mutation when paired with uniform crossover for DS1 and arithmetic crossover for DS7, using the parameters shown in Table III , as before. The set coverage results for these experiments illustrate that Gaussian mutation gives better results than uniform mutation.

The evaluation results show that Gaussian mutation has a slight tendency to outperform uniform mutation in both small and large data sets. Hence Gaussian mutations are the most effective mutation operators in the research; therefore, they should be considered for further investigation.

D. Lifting Allocation Mutation

The mutation operators defined and applied so far are limited in the extent of the changes that they make. For small

values of P_m, λ_P and λ_A, there is very little perturbation to the chromosome, which is unlikely to have a significant effect on the overall cost. However, larger values of P_m, λ_P and λ_A perturb the chromosome in an uncoordinated fashion, potentially losing any beneficial structure in the solution. To address this, a modified mutation operator was proposed that applies changes to all the bandwidths allocated to all the houses in a coordinated fashion, adding the same random value to them all. *Lifting allocation mutation* is performed by adding a single normally distributed random value (delta) to every house. Lifting is performed with probability P_r, otherwise the previously defined Gaussian mutation is applied. The P_r represents the probability rates of 0.3, 0.5 and 0.9. The lifting mutation within NSGA-II was applied to the data sets. The set coverage shows good performance with lifting mutation for DS1 with uniform crossover and Gaussian mutation compared to the data set without lifting allocation. From the set coverage result of E6.1 it was observed that the lifting allocation with P_r of 0.5 and 0.9 shows better performance than P_r (0.3). For further investigation of DS1 the lifting allocation with P_r of (0.9) was then used. Applying lifting mutation to DS7 with arithmetic crossover and Gaussian mutation also showed an improvement. The result of E6.3 illustrates that lifting allocation with the Pr of 0.3, 0.5 and 0.9 outperformed the data set without lifting allocation, giving set coverage of 40%, 45% and 40%, respectively. The lifting allocation with Pr (0.5) outperformed the lifting allocation with Pr (0.3 and 0.9). The potential of a good experimental result from the lifting allocation was perceived with Pr (0.5), presenting a good diversity of solutions in the search space (see Figure 2) which improved the performance of allocation with arithmetic crossover and Gaussian mutation.

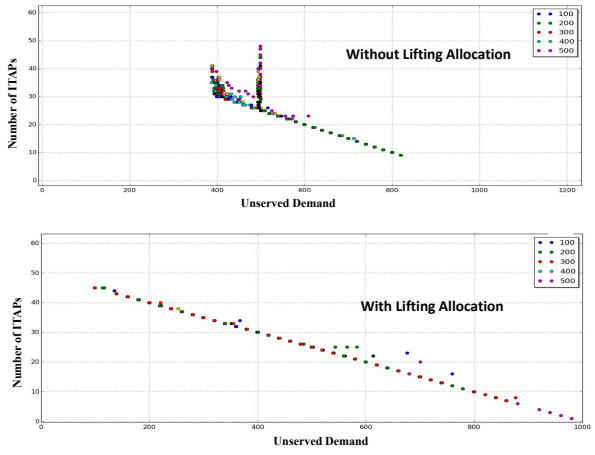

Fig. 2. Diversity of solution in the search space without and with Lifting Allocation for DS7 of E6.3

In experiment E6.6 data set DS7 was investigated to demonstrate the set coverage for lifting allocation applied with arithmetic crossover and Gaussian mutation. The lifting allocation shows good performance compared to other cases. The lifting allocation for the Pr (0.5) did better than other rates. For further investigation of DS7 of E6.3 and E6.6 the lifting allocation with Pr (0.5) was used. The lifting allocation mutation showed a marked improvement in the diversity of the solutions in the search space of the sampled data sets.

E. Aggressive Placement Mutation

The previous section addressed the limitations of mutation for bandwidth allocation by applying a "lifting" adjustment to all houses. Here, modifying the mutation of placement genes by applying an "aggressive mutation" was proposed. Like lifting, aggressive identically mutates not a single gene but a whole chromosome with mutation probability λ_A/N, thereby making greater changes possible. The selected mutating gene of the placement is swapped with the K value. The $K = 2, 5, 10$ value is a randomly chosen number from the set of 2, 5 and 10. The value of K used for aggressive placement mutation was compared in an experiment with Gaussian placement mutation, to see the effect of the changes that the aggressive mutation can make. The set coverage of aggressive placement mutation of value $K = (2, 5)$ in DS1 of E6.1 showed a great improvement compared to the Gaussian placement mutation with lifting allocation. However, the aggressive placement mutation of value $K = 10$ showed no improvement over the Gaussian with lifting allocation. The set coverage of DS7 of E6.3 shows the aggressive placement mutation performance of value $K = (2, 5)$ compared to Gaussian mutation with lifting allocation that has the same improvement value. The aggressive placement mutation of value $K = 10$ shows a small improvement. The set coverage of DS7 of E6.6 demonstrates a greater improvement for aggressive placement mutation than Gaussian placement mutation with lifting allocation. The aggressive mutation of value K ensured a better performance than Gaussian placement mutation. Thus, the aggressive placement mutation of value K showed an improvement over the Gaussian placement mutation in all data sets.

F. Mutation Rate

The mutation applied in sections V and VI is controlled by a number of parameters. P_m controls whether an individual chromosome is mutated (otherwise, it is left unchanged), while P_r, λ_P, and λ_A, control the gene mutation in placement and allocation. Since the mutation rate can be very problem-specific, it is better to run experiments with several rates to see which rate maintains the greatest diversity in the population. Using too high a mutation rate will increases the diversity in the search space, but hinders convergence. At the same time, using too small a mutation rate may result in premature convergence (leading to local optima instead of a global optimum). In other words, too high a mutation rate reduces the search ability of NSGA-II to simple random sampling, while too small a mutation rate almost always fails, resulting in a local optimum due to the lack of diversity in the search space. In the previous experiments a mutation rate of 0.1 was used and then tests with mutation rates of 0.5 and 1.0, with lifting

TABLE IV
SUMMARY OF THE USED OPERATORS AND PARAMETERS VALUE WITH THE GENETIC ALGORITHM

Data Set	Crossover	Mutation	Lifting Allocation	Aggressive Mutation	Mutation Rate
DS1	Uniform	Gaussian	0.9	2 and 5	0.5
DS7	Arithmetic	Gaussian	0.5	2 and 5	0.5

allocation were carried out, comparing these with the previous mutation rate. The set coverage show that the mutation rate 0.5 outperformed the mutation rate of 0.1 in the data sets of experiments (E6.1, E6.3 and E6.6). The mutation rate 1.0 showed a slight improvement only in the large data sets of E6.3 and E6.6. The experimental results demonstrate that the set coverage for the mutation rate of 0.5 outperformed that of the mutation rate of 0.1, which indicates the effectiveness of the former over the latter. The best mutation rate seems to be in the range of 0.5 of population size 32.

To sum up the experimental results obtained by the data sets with the suggested operators and parameters that were investigated with, see in Table IV the conclusion of the GA progress.

To test the algorithm and to evaluate the performance of the NSGA-II approach, the set coverage of NSGA-II and the weighted sum approach are compared for the data set samples of DS1, DS7 and DS7/E6.6, using the parameters of Table IV plus Gaussian mutation. The evaluation result shows that NSGA-II algorithm outperforms the weighted sum approach for data sets DS1 and DS7. The non-dominated solutions of the weighted sum are clustered together while the solutions of NSGA-II are spread out in the search space; for example, in DS7 the maximum and minimum numbers of ITAPs in the weighted sum were 50 and 20, but in NSGA-II they were 49 and 0 respectively. This shows that the crowding distance in NSGA-II is higher and gives better results than the weighted sum.

VII. CONCLUSION

In this paper an evolutionary multi-objective optimization algorithm was presented to solve the problem of optimizing WMN infrastructure placement, defining the GA which involves applying crossover and mutation operators on two individuals. The NSGA-II algorithm creates a child population from its parent population using fast non-dominated crossover and mutation. Several initial solutions with different operators and parameters were investigated, to ensure the presence of diversity, aiming to identify the best operator and parameters to use as part of the comprehensive solution methodology. The literature mentions a great variety of crossovers; the ones illustrated here were *Arithmetic* and *Uniform* crossovers. The set coverage comparison of crossover indicated that the uniform crossover outperformed the arithmetic crossover in small data sets such as DS1 and DS1A, while the arithmetic crossover outperformed the uniform crossover in big data sets such as DS7 and DS7A. The experimental results on the data sets samples indicate that the Gaussian mutation outperformed uniform mutation and was effective as a genetic operator. However, the results for arithmetic crossover and Gaussian mutation dramatically improved when used in combination with a lifting allocation of 0.5 probability. Further investigation of the potential of these operators is suggested. To change the mutation of the placement genes an aggressive placement mutation was applied; the results of this experiment demonstrated the performance of aggressive placement mutation compared to Gaussian placement mutation. A random mutation was applied to one or more genes for the earlier test of instances the mutation rate was set to 0.1 and then mutation rates of 0.5 and 1.0 were applied. The experimental result of the mutation rate at 0.5 outperformed the mutation rate at 0.1 and 1.0. The greatest performance improvements of mutation were obtained by using a mutation rate of 0.5 with a gene probability mutation of λ_P/N and λ_A/M. To conclude this paper, the NSGA-II algorithm results were compared with the results of using the weighted sum approach. the set coverage result showed that the NSGA-II outperformed the weighted sum approach for the sampled data sets. This indicates the effectiveness and good performance of NSGA-II. It was observed from the experiments that the NSGA-II algorithm performed generally better than the weighted sum approach in terms of diversity and quality from the approximation of the Pareto front.

REFERENCES

[1] O. Rifki and H. Ono, "A survey of computational approaches to portfolio optimization by genetic algorithms," in *18th International Conference Computing in Economics and Finance*, 2012.

[2] M. Camelo, C. Omana, and H. Castro, "Qos routing algorithms based on multi-objective optimization for mesh networks," *IEEE Latin America Transactions*, vol. 9, no. 5, pp. 875–881, 2011.

[3] N. Srinivas and K. Deb, "Muiltiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary computation*, vol. 2, no. 3, pp. 221–248, 1994.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[5] B. Aoun, R. Boutaba, Y. Iraqi, and G. Kenward, "Gateway placement optimization in wireless mesh networks with qos constraints," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 11, pp. 2127–2136, 2006.

[6] R. Chandra, L. Qiu, K. Jain, and M. Mahdian, "Optimizing the placement of internet taps in wireless neighborhood networks," in *Network Protocols, 2004. ICNP 2004. Proceedings of the 12th IEEE International Conference on*. IEEE, 2004, pp. 271–282.

[7] L. Nawaf, S. M. Allen, and O. Rana, "Internet transit access point placement and bandwidth allocation in wireless mesh networks," in *Computing and Communication Workshop and Conference (CCWC), 2017 IEEE 7th Annual*. IEEE, 2017, pp. 1–8.

[8] G. S. Ladkany and M. B. Trabia, "A genetic algorithm with weighted average normally-distributed arithmetic crossover and twinkling," *Applied Mathematics*, vol. 3, no. 10A, pp. 1220–1235, 2012.

[9] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proceedings of the 3rd International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1989, pp. 2–9. [Online]. Available: http://dl.acm.org/citation.cfm?id=645512.657265

[10] R. Tinós and S. Yang, "Evolutionary programming with q-gaussian mutation for dynamic optimization problems," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE, 2008, pp. 1823–1830.

APPENDIX

List of abbreviations used in this paper.

b_h	Bandwidth allocated to each house
Cap_e	Edge Capacity
Cap_h	House Capacity
Cap_i	ITAP Capacity
DS	Dataset
EA	Evolutionary algorithm
f_D	Unsatisfied Demand solution
f_I	Number of ITAPs solution
f_U	Unfairness solution
GA	Genetic Algorithm
ITAP	Internet Transit Access Point
MOEA	Multi-objective Evolutionary Algorithm
MOO	Multi-Objective Optimization
M	Number of Houses
N	Number of ITAP location
NSGA-II	Non-Dominated Sorting Genetic Algorithm
P	Population
P_m	Mutation Rate
P_r	Probability Rate
QoS	Quality of service
VoIP	Voice over Internet Protocol
w_h	House Demand
WMN	Wireless Mesh Network
$x_{e,h}$	Flow from edge to house
y_i	Number of ITAPs installed at location i