Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

*Digital Object Identifier 10.1109/ACCESS.2017.DOI*

# Comparison of metaheuristic algorithms for interface-constrained channel assignment in a hybrid Dynamic Spectrum Access – Wi-Fi infrastructure WMN

**NATASHA ZLOBINSKY[1], DAVID L. JOHNSON[2], AMIT K. MISHRA[3], (Senior Member, IEEE) AND ALBERT A. LYSKO[4,1], (Senior Member, IEEE)**

[1]Department of Computer Science, University of Cape Town, Rondebosch, 7701 South Africa (e-mail: natzlob@gmail.com)
[2]Department of Computer Science, University of Cape Town, Rondebosch, 7701 South Africa, (e-mail: david.lloyd.johnson@gmail.com)
[3]Department of Electrical Engineering, University of Cape Town, Rondebosch, 7701 South Africa, (e-mail:akmishra@ieee.org)
[4]NextGen Enterprises and Institutions, Council for Scientific and Industrial Research, Pretoria, 0001 South Africa, (e-mail:alysko@csir.co.za)

Corresponding author: Natasha Zlobinsky (e-mail: natzlob@gmail.com).

**ABSTRACT** In this work, we evaluate the application of four different metaheuristic optimisation algorithms for solving the channel assignment problem in a multi-radio multi-channel Wireless Mesh Network (WMN) using Dynamic Spectrum Access (DSA). The work advances a near optimal channel assignment in a WMN that uses DSA by applying soft computing methods. While CA in a WMN is well-studied, and CA for secondary user cognitive radio networks has also been studied in the literature, CA for our specific scenario of an infrastructure DSA-WMN is novel. This scenario poses new challenges because nodes are spread out geographically and so may have different allowed channels and experience different levels of external interference in different channels. A solution must meet two conflicting requirements simultaneously: 1) to avoid interference within the network and with external interference sources, and 2) maintain connectivity within the network; all while staying within the radio interface constraint, i.e., only assigning as many channels to a node as it has radios. Our method is unique in that it is protocol-agnostic, being able to avoid interference from external sources that use different protocols and standards. We present a novel algorithm, used alongside the metaheuristic optimisation algorithms, which ensures the feasibility of solutions in the search space. Average Signal to Interference and Noise Ratio ($SINR$) over the network is used as the performance measure, with the goal of optimisation being to find the highest average $SINR$. This is a more realistic performance measure than the binary on/off conflict-based measures most common in the literature. Our energy-based method also has the unique advantage that it is protocol-agnostic, being able to avoid interference from external sources that use different protocols and standards. The algorithms that are compared in this work are Simulated Annealing (SA), the Genetic Algorithm (GA), Particle Swarm Optimisation (PSO), and Differential Evolution (DE). These algorithms were evaluated through the use of simulation in Network Simulator 3. Various parameters were experimented with for each of the employed algorithms. The resultant best set of parameters was used for the comparison of the four metaheuristic algorithms. It was found that the population-based algorithms (PSO, GA, and DE) all perform satisfactorily for this problem, although DE is superior to the others. SA can give acceptable solutions, but performs poorly in comparison to the population-based algorithms. The paper also considers the computational complexity of the methods. It is found that SA and DE perform well in this regard.

**INDEX TERMS** Wireless Mesh Networks, WMN, Dynamic Spectrum Access, DSA, optimisation, channel assignment, Simulated Annealing, Genetic Algorithm, Particle Swarm Optimisation, PSO, Differential Evolution, NS3, simulation

## I. INTRODUCTION

DYNAMIC Spectrum Access (DSA) has recently been gaining traction once again, as regulatory bodies around the world are opening up the spectrum bands that were formerly reserved for licensed users, for opportunistic use by other users. Recent examples are Citizens Broadband Radio Service (CBRS) Spectrum Access System [1], Automated Frequency Coordination in Wi-Fi 6E [2] and Dynamic Spectrum Sharing for 5G and LTE [3]. A more traditional example is Television White Spaces (TVWS). Most of the associated spectrum bands require (or will require) the use of spectrum databases for secondary users to acquire access to channels within the bands. Meanwhile, Wireless Mesh Networks (WMNs) have proven their usefulness in extending Internet access from a gateway node to a wider area [4]–[6], especially in rural areas or informal settlements where Internet connectivity infrastructure is not reliable. Bringing together the technologies of DSA and WMNs has not yet enjoyed much attention in the literature, even though their combination can be very advantageous. DSA-WMNs can aid in bringing connectivity to the unconnected, unreliably connected, or underserved.

This novel type of hybrid DSA-WMN comes with new challenges and avenues for research. One particularly difficult problem that arises in such DSA-WMNs is Channel Assignment (CA). The CA problem is already NP-complete[1] [8]. Several factors add to the complexity of this problem in the case of DSA-WMNs. These factors include limited channel availability evaluated by the spectrum database to prevent interference with licensed users, the fact that different nodes may have different allowed channels and experience different channel conditions and external interference, and interference within the network. In addition to the requirement to minimise interference, the WMN brings the contrasting need to maintain connectivity within the network. This connectivity requirement, while preventing interference or high contention for the same channel, complicates the traditional DSA cognitive radio CA problem. Our work presents and compares the performance of Simulated Annealing (SA) and population-based methods (Genetic Algorithm, Particle Swarm Optimisation, and Differential Evolution) for the CA problem in a hybrid Wi-Fi-DSA-WMN. To our knowledge, the application of the above-mentioned methods in the Wi-Fi-DSA-WMN CA domain is one of the novelties of our work.

The rest of this paper is organised as follows: In Section II, we offer background on DSA, Channel Assignment in WMNs, and the metaheuristic optimisation algorithms employed in this work. In Section III, we present related studies. The problem is formulated in detail in Section IV. We present our methodology in Section V, including our new algorithm for ensuring solution feasibility. Finally, our results are provided in Section VI before concluding with Section VII.

[1]"NP" stands for "non-deterministic polynomial time". The class NP is the class of problems for which a given proposed solution for a given input can be verified by a polynomial-time algorithm but, thus far, no polynomial-time algorithm has been found for solving members of this class of problems. The class of problems consisting of the "hardest" problems in NP are called NP-complete problems [7].

## II. BACKGROUND

### A. DYNAMIC SPECTRUM ACCESS

It was found that large parts of the radio spectrum remain unused while being licensed to certain users, creating an artificial spectrum scarcity. This set the stage for Dynamic Spectrum Access to emerge as a way for the radio frequency spectrum to be used more efficiently. DSA refers to techniques whereby wireless frequency bands can be shared between the primary (licensed and thus legally protected from interference) users of the spectrum and secondary (unlicensed) users. DSA is enabled by Cognitive Radios (CRs) through spectrum sensing and/or the use of Geolocation Spectrum Databases (GLSDs). While practical spectrum sensing still remains in the research stage, GLSD-based approaches have received wide acceptance and practical application, e.g., [9]–[11]. By applying these methods, radios can adjust their spectrum access and use according to current environmental conditions while ensuring that Primary Users (PUs) or incumbents are protected from harmful interference.

TVWS is one such band in which DSA based on GLSD technology is used. TVWS refers to the unused portions of the spectrum in the 470-694 MHz range traditionally licensed to TV broadcasting. (This TV broadcasting band is particular to Europe; other countries may have slightly different but mostly overlapping bands). Secondary Users (SUs) have been allowed to access this spectrum by a number of national regulatory bodies, including the Federal Communications Commission (FCC) in the United States of America, the Office of Communications (Ofcom) in the United Kingdom, and the Independent Communications Authority of South Africa (ICASA) in South Africa. Most regulations require the use of a GLSD to ensure compliance and protection of PUs.

Another DSA band is the Citizens Broadband Radio Service (CBRS). The CBRS is a band of spectrum in the 3.5 GHz range that was recently opened for sharing with incumbents for commercial use in the United States [1]. Service providers can deploy networks in this band without requiring spectrum licenses. Access is divided into three tiers: incumbent access, priority access, and general authorised access. CBRS uses a Spectrum Access System (SAS), which grants requests by SUs to access channels in the band, using a database of CBRS radio base stations, similar to the GLSD in TVWS.

In order to minimise interference with satellite links, Wi-Fi 6E is set to use an Automated Frequency Selection (AFC) system, as the 6 GHz band has been opened up for unlicensed use by either low-power indoor Access Points (APs) or standard-power outdoor Wi-Fi APs [2]. The AFC system will also use a spectrum database to coordinate spectrum use among users. To obtain available channels and request access, APs must consult the registered database of an AFC provider before starting to transmit.

Our work can be extended to any and all of these DSA bands or any future bands, and so we expect it to become increasingly useful in time.

### B. CHANNEL ASSIGNMENT IN WIRELESS MESH NETWORKS

Different channels may be optimal for use by different nodes in a WMN, and different channels may experience differing levels of external interference and utilisation. So, different channels would be good choices for different nodes placed in different locations. WMN nodes also have to share a common channel to be able to communicate. Thus, the problem of assigning channels optimally is an important and difficult one in WMNs. Proper channel assignment can improve spectral efficiency while also minimising interference, increasing throughput, and maintaining connectivity in the network. The CA problem is well-known to be NP-complete since it is, in essence, a graph-colouring problem [8], [12], [13]. Despite considerable attention to the question in the literature, there is currently no universally good solution to this CA problem in WMNs. Additionally, substantial opportunities for further research and improvements exist in the case of DSA- and CR-using WMNs.

A good CA must:

- minimise the interference experienced within the network and from external interference sources,
- maximise $SINR$,
- maintain connectivity along necessary paths between nodes, and
- improve overall network capacity and performance.

CA and routing are interdependent. Optimal channels change with the routes selected by the routing algorithm, which links are used, and where the bottleneck links are in the network. Conversely, routing depends on the capacity of links, which is dependent on the conditions of the channels assigned to those links. However, every new packet that is offered to the network may need new routing. Hence, routing is a fast-changing process. On the other hand, changing channels is usually a slower process since it takes on the order of seconds for a Network Interface Card (NIC) to find an alternative channel, switch its channel, and reconnect the link for all current hardware. A CA algorithm that could take into account parameters that change more quickly, e.g., link quality, traffic conditions, and routes, would be more adaptive to changing conditions in the network. However, it would likely result in loss of connectivity more often. Overall, this degrades the network's performance. A too rapidly changing CA would also result in the routing having to re-adapt often and may result in a race condition between routing and CA, causing the network to become unstable. Also, frequently propagating the required monitoring and control information through a large network would significantly consume the network's bandwidth and reduce the achievable goodput, as is the case in WMNs [14], [15]. For these reasons, routing must be taken into account in a CA, but the CA must be a more global longer-term solution than routing. An optimal CA algorithm would have to take spectrum measurements over time covering a variety of routing configurations and use this information as an input.

CA algorithms might increase the overhead in the network due to the communication required to distribute channel information, exchange decision-making information, and update the state of the network. We do not consider the effect of overhead on network performance in this work. It is kept as possible future work.

### C. METAHEURISTIC ALGORITHMS FOR OPTIMISATION

We give some brief background on the metaheuristic stochastic optimisation algorithms employed in this work. We have selected these algorithms because they are some of the most well-known, tried-and-tested and readily available algorithms [16]. This means these algorithms are easier to implement in a real network. It also means that our experiments can be replicated readily using the same algorithms, perhaps in other coding languages or with other simulation frameworks.

#### 1) Simulated Annealing

Simulated Annealing (SA) is a probabilistic search heuristic used in optimisation problems with complex, often discrete, search spaces. It is based on, and analogous to, the physical process of annealing (of a metal, for example) in statistical mechanics, whereby atoms are cooled in a specific slow way until reaching the state of minimum energy [17]. The algorithm starts with the system in a certain arbitrary configuration or state, i.e., a solution, and then it computes the "energy", which is the value of the objective function or cost of that solution at that iteration. From there, a new candidate neighbour solution is generated by applying a slight alteration to the system state. Then, the candidate solution's cost value is computed and compared to the original cost. The candidate solution is either accepted or rejected based on its cost value. It is always accepted if the cost has improved ("energy" has decreased). The candidate is accepted probabilistically if the new solution is worse, with the probability based on the difference in cost between the new candidate solution and the old solution, as well as the temperature parameter. The accepted solution is then the starting point for the next iteration.

The temperature parameter is related to how likely the algorithm is to choose a worse solution than the current one to prevent the algorithm from getting stuck in a local minimum. The temperature must initially be set to a higher value and decreased every iteration according to a defined cooling function, the choice of which is up to the implementer. Some examples are exponential multiplicative cooling, logarithmic multiplicative cooling, and linear multiplicative cooling [18]. The aim of SA is always to find/converge to the lowest "energy" configuration, which is the solution with the lowest cost. As the number of iterations increases, the probability of finding the true optimal solution increases. The process of generating a new neighbour solution and accepting or reject-

ing the solution continues until predetermined termination criteria are met. For example, a specified number of iterations or acceptable running time is reached, and a satisfactory solution has been settled on. Certain tests and rules-of-thumb can be followed to determine whether to stop or continue with the algorithm or estimate the convergence time, e.g., the Geweke test [19].

#### 2) Genetic Algorithm

The Genetic Algorithm (GA) is a well-known metaheuristic based on the evolution of genes through generations. In GA, the fittest individuals are selected as parents and the fittest genes are carried on in future generations. The selected parents reproduce and occasional mutation occurs to the genes. The required elements of a GA are:

- a fitness function (optimisation objective function);
- a population of chromosomes, also called genomes (an encoding for solutions in the solution search space);
- a selection method by which parents of the next generation are selected;
- a crossover method by which parents reproduce to create the next generation; and
- a mutation method by which random changes are introduced to the chromosomes, preventing premature convergence to local minima.

The algorithm starts by generating a starting population at random from the search space and applies selection, crossover, and mutation every iteration. This cycle continues until termination. Termination may occur when the fitness value of the chromosome with the best value thus far, stays the same for a certain number of iterations, or after an acceptable predefined total number of generations is reached. One of several parent selection methods may be used. A popular method is Roulette Wheel selection, where each chromosome in the current generation is given a probability of being selected that is proportional to its fitness. This method is vulnerable to causing premature convergence. Linear Rank selection tries to prevent the situation observed in Roulette Wheel selection, where a single solution dominates and causes premature convergence. Linear Rank selection instead ranks individuals according to their inverse fitness and then bases the probability of selection on the rank rather than the actual fitness value. The highest fitness solutions are given the highest value rank (lowest rank). For example, out of ten solutions, the highest fitness will have rank position 10 (not 1), so it is the most likely to be chosen, and the lowest fitness will have rank 1, having the lowest probability of being selected. Crossover may be single-point or multi-point. In crossover, the selected parent chromosomes are subdivided into sections and the sections swapped out to generate new combinations of genes.

#### 3) Differential Evolution

Differential Evolution (DE) is a member of the group of evolutionary algorithms. It was initially developed to deal with continuous variable problems using evolutionary methods [20]. However, it has also been used in discrete settings [21].

In DE, an initial population of size $NP$ is chosen at random, similarly to GA. Each individual in a population is represented by $x_{i,G}\ \forall\ i = 1, 2, ..., NP$ and $G$ denotes the generation. The mutation operation, which occurs at every iteration, is different from other evolutionary algorithms. The *mutant* vector is generated using the weighted difference between two other randomly chosen members of the population and adding that to a third randomly chosen member of the population, not equal to either of the previous two. In the crossover stage, the parameters of the mutant vector are again mixed with the parameters of another predetermined vector, the *target* vector. This is done in order to increase the diversity of the perturbed parameter vectors. This addition is called "mixing" and results in a *trial* vector.

A selection operation replaces the target vector, or the parent, with the trial vector if the latter yields a lower cost value than the parent. Hence, the fitter offspring now becomes a member of the newly generated population. These iterations continue until one of the termination criteria is reached.

The DE parameters to be selected are the population size $NP$; $F \in [0, 2] \subset \mathbb{R}$, a constant factor that controls the amplification of the difference vector; and $CR$, the crossover rate. Small values of $F$ will result in smaller mutations (with less variance), resulting in slower convergence of the algorithm. Larger values of $F$ may cause overshoot and convergence to the wrong value.

The steps involved in DE are detailed below.

##### a: Mutation

For each target vector $x_{i,G}$ in the population, randomly select three other individuals from the population: $r_1, r_2, r_3 \in \{1, 2, .., NP\}$, ensuring that

$$i \neq r_1 \neq r_2 \neq r_3.$$

Mutant vector $v_{i+1,G}$ is then computed using equation (1).

$$v_{i,G+1} = x_{r_1,G} + F \cdot (x_{r_2,G} - x_{r_3,G}) \tag{1}$$

##### b: Crossover

The trial vector

$$u_{i,G+1} = (u_{i1,G+1}, u_{i2,G+1}, u_{i3,G+1}, ..., u_{iD,G+1})$$

is formed, where $D$ is the number of dimensions, according to equation (2).

$$u_{ij,G+1} = \begin{cases} v_{ij,G+1} & \text{if } rand(j) \leq CR \text{ or } j = rand(i) \\ x_{ij,G} & \text{if } rand(j) > CR \text{ and } j \neq rand(i) \end{cases} \tag{2}$$

$\forall\ j = 1, 2, ..., D$

where

$rand(j)$ is the $j^{th}$ outcome of a random binary number in $[0, 1]$, or $rand(j) \sim U(0, 1)$, depending on the implementation;

$rand(i)$ is randomly chosen from $[1, 2, ..., D]$;
$CR$ is the crossover constant $\in (0, 1) \subset \mathbb{R}$.

c: Selection

To decide whether or not it should become a member of generation $G + 1$, the trial vector $u_{i,G+1}$ is compared to the target vector $x_{i,G}$ as follows: If the trial vector $u_{i,G+1}$ yields a smaller cost function value (which is better, assuming the goal is minimising the cost) than $x_{i,G}$, then $x_{i,G+1}$ is set to $u_{i,G+1}$; otherwise, the old value $x_{i,G}$ is retained in the next generation, i.e.

$$x_{i,G+1} =: \begin{cases} u_{i,G+1} & \text{if } f(u_{i,G+1}) < f(x_{i,G}) \\ x_{i,G} & \text{otherwise} \end{cases} \quad (3)$$

d: Variants

A naming convention of $DE/x/y/z$ has been adopted. $x$ denotes the vector to be mutated, which can be $rand$ (random), or $best$ (chromosome with the lowest cost). $y$ denotes the number of difference vectors considered for the mutation of $x$, and $z$ is the type of crossover in use. In the previous example in Section II-C3b, this was $binomial$, since it was done by independent binomial trials. The method we have described would be termed $DE/rand/1/bin$. Some other variants are:

$DE/best/1$:

$$v_{i,G+1} = x_{best} + F \cdot (x_{r_1,G} - x_{r_2,G}) \quad (4)$$

$DE/rand/2$:

$$v_{i,G+1} = x_{r_1,G} + F \cdot (x_{r_2,G} - x_{r_3,G}) + F \cdot (x_{r_4,G} - x_{r_5,G}) \quad (5)$$

$DE/best/2$:

$$v_{i,G+1} = x_{best,G} + F \cdot (x_{r_1,G} - x_{r_2,G}) + F \cdot (x_{r_3,G} - x_{r_4,G}) \quad (6)$$

$DE/current-to-best/2$:

$$v_{i,G+1} = x_{i,G} + F \cdot (x_{best,G} - x_{i,G}) + F \cdot (x_{r_1,G} - x_{r_2,G}) \quad (7)$$

where

$$i \neq r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5.$$

4) Particle Swarm Optimisation

a: Overview

Particle Swarm Optimisation (PSO) is another stochastic population-based search algorithm. It was inspired by the behaviour of animals in nature, such as flocks of birds or schools of fish, which work in groups to locate desirable positions [22]. A desirable position might have, for example, good food sources. PSO is driven by the assumption that each individual member of a swarm benefits from the experience of other members of the swarm to the overall advantage of the group.

There are two main operations per iteration: First, the velocity of every particle is updated. Second, the position of every particle is updated according to the calculated velocity. The canonical velocity calculation is shown in equation (8), and the position update is shown in equation (9).

$$\begin{aligned} \boldsymbol{v_i}(t+1) = \omega \boldsymbol{v_i}(t) + c_1 \boldsymbol{r_1}(t) \times (\boldsymbol{y_i}(t) - \boldsymbol{x_i}(t)) \\ + c_2 \boldsymbol{r_2}(t) \times (\hat{\boldsymbol{y}}(t) - \boldsymbol{x_i}(t)) \end{aligned} \quad (8)$$

$$\boldsymbol{x_i}(t+1) = \boldsymbol{x_i}(t) + \boldsymbol{v_i}(t+1) \quad (9)$$

where
$t$ is the iteration counter;
$r \sim U(0, 1)$ is a (pseudo)-random number selected in the range 0 to 1 and $r_1$ and $r_2$ are generated anew for every dimension;
$\boldsymbol{v_i}(t)$ is the velocity of particle $i$ at iteration $t$;
$\boldsymbol{x_i}(t)$ is the position of particle $i$ at iteration $t$;
$\boldsymbol{y_i}$ is the position of the particle $i$ found so far with the best fitness (lowest cost);
$\hat{\boldsymbol{y}}$ is the position of any particle in the swarm found so far with the best fitness (lowest cost)
$\omega$ is a coefficient causing inertia in the movement (weighting towards the previous velocity);
$c_1, c_2$ are named cognitive and social coefficients, respectively, altering the relative weight of the particle's own memory and that of the swarm.

(We can think of the velocity as a displacement in constant time, to ease the discomfort of inconsistent quantities when adding a velocity to a position). Each of these calculations is done per dimension, the number of which is not limited. The inertia weight $\omega$ determines the balance between local and global search. The smaller $\omega$ is, the more the algorithm behaves like a local search, searching around its current position. In this case, the second and third terms of equation (8) dominate. The larger $\omega$ is ($> 1.2$), the more expanded the search space becomes, and the algorithm searches more globally [23], [24].

Several variants of PSO are based on slight changes to the velocity equation. There is also the "bare bones" PSO in which the position is changed by using a normal distribution centred around the mean of the personal best and group best position [25].

b: Naming and Variants

Several variants on the velocity computation have been put forward. Some of these are:

- *Variant 2* (equation (10)), where the same random variable $r_1$ is used for both the social and cognitive components.

$$\begin{aligned} \boldsymbol{v_i}(t+1) = \omega \boldsymbol{v_i}(t) + c_1 \boldsymbol{r_1}(t) \times (\boldsymbol{y_i}(t) - \boldsymbol{x_i}(t)) \\ + c_2 \boldsymbol{r_1}(t) \times (\hat{\boldsymbol{y}}(t) - \boldsymbol{x_i}(t)) \end{aligned} \quad (10)$$

  A new $r_1$ is generated for every dimension.
- *Variant 3* (equation (11)), where the same two random variables are used for all dimensions.

$$\begin{aligned} \boldsymbol{v_i}(t+1) = \omega \boldsymbol{v_i}(t) + c_1 r_1(t) \times (\boldsymbol{y_i}(t) - \boldsymbol{x_i}(t)) \\ + c_2 r_2(t) \times (\hat{\boldsymbol{y}}(t) - \boldsymbol{x_i}(t)) \end{aligned} \quad (11)$$

A new $r_1$ and $r_2$ are generated once, and the same values are used for all dimensions.

- *Variant 4*, with only one random variable used for all quantities. Variant 4 is the same as Variant 3, except that $r_1$ is only generated once and reused for every dimension.
- *Variant 5*, where the constriction coefficient affects the whole calculation and not just the previous velocity (equation (12)). In this variant, $\omega$ is sometimes written as $\chi$, since it represents a slightly different quantity from the inertia coefficient.

$$\boldsymbol{v_i}(t+1) = \omega \times (\boldsymbol{v_i}(t) + c_1 r_1(t) \times (\boldsymbol{y_i}(t) - \boldsymbol{x_i}(t)) + c_2 r_2(t) \times (\hat{\boldsymbol{y}}(t) - \boldsymbol{x_i}(t))) \tag{12}$$

Constriction weight $\omega$ (or $\chi$) applies to all components.

- *Variant 6* is called *Fully Informed Particle Swarm* (FIPS) and shown in equation (13). FIPS is different from the other variations as each particle is not just affected by itself and the best neighbour, or the best particle in the swarm. FIPS includes effects from all the particles $k$ in the swarm or neighbourhood of size $K$. The random number $r(t) \sim U(0, c_1 + c_2)$.

$$\boldsymbol{v_i}(t+1) = \omega \boldsymbol{v_i}(t) + \frac{1}{K} \sum_{k=1}^{K} r(t) \times (\boldsymbol{y_k}(t) - \boldsymbol{x_i}(t)) \tag{13}$$

The PSO was originally conceptualised as a method in continuous space. It has, however, been used liberally in a discrete form, e.g., [26]–[29], with different ways of discretising the values and operators, normally by simple rounding, or by introducing a penalty function.

## III. RELATED WORK: METAHEURISTIC ALGORITHMS FOR CA IN DSA WMNS

While the channel selection and assignment problems may appear to be well studied, there is no existing work before ours that applies metaheuristic optimisation algorithms for CA to an infrastructure WMN using DSA methods. To the best of our knowledge, ours is also one of few works to use the $SINR$ perceived by the nodes for CA in a WMN. In contrast, it is common in the literature to use simple binary conflict-based interference objectives, neglecting the maintenance of connectivity requirement, and using unrealistic interference and channel models.

Chowdhury and Akyildiz presented the concept of a cognitive WMN with DSA [30]. However, their envisaged scenario is very different from ours. In our work the WMN is the backbone and the WMN links are using the DSA spectrum, while the WMN nodes perform the sensing; whereas, in [30], the clients do the sensing and clients form clusters, while the links between the mesh nodes are formed out of band through communication on a different dedicated channel. The actual mechanism for forming these WMN links is not outlined in their work. The authors also do not attempt to optimise the CA, but concentrate on the sensing problem. The channel switching algorithm is only for shifting some clusters from the secondary band into the primary band.

DSA WMNs are also approached by Xin et al. [31]. They present a distributed CA algorithm for a DSA WMN. This solution does not attempt to optimise the assignment and does not use any soft computing methods. They also consider only single-radio nodes, while our scenario considers nodes with more than one radio. There are practical questions around their work. For example, we consider their assumption that a node randomly switching to a channel is likely to find a node to link with on that channel unrealistic. Xin et al. also make the assumption that the accessible channel list is the same for all nodes in the network, whereas we consider the more realistic case where the available channel list is different for each node. The case of a multi-radio multi-channel network as SUs coexisting with PUs is addressed by Qin et al., using Lyapunov optimisation of throughput and average delay [32]. One of their considered scenarios is a multi-hop network. The network scenario in this work is quite simple as there are only 5 source-destination pairs and a maximum of 15 SU nodes. In addition, interference within the network is not adequately addressed.

A CA algorithm is given for an infrastructure WMN using Wi-Fi spectrum by Ramachandran et al. [33]. Their CA is built on a novel interference estimation scheme. In this interference estimation method a packet capture interval is used to identify the number of MAC addresses external to the network. These MAC addresses identify the number of interfering devices. The packet capture is also used to gather the channel utilization of the interfering devices. The two lists of devices and channel utilization are ranked and merged by averaging the quantities to form the interference estimation. This interference estimation technique is fairly easy to implement in real nodes, but is not a realistic model for interference, since it is based on the number of conflicts and does not include cumulative interference effects. It can also only identify other users of the same technology (Wi-Fi) and would not work in the case of mixed technologies in the same spectrum band. The required packet capture period causes a temporary disruption to transmissions from each capturing radio of 36 s, which is not ideal. The researchers also claim that this interference estimation is done every 5 minutes. This would be a significant disruption to the network and would likely cause a bad user experience. However, the pertinent aspect of this work in relation to ours is that the authors address interference between the WMN itself and other co-located wireless networks (as well as within the network itself). This mirrors our case of interference being experienced from external SUs in the vicinity of our network, even though [33] only considers interference between Wi-Fi devices. This contrasts with our scenario where the external SUs may be using a different technology or standard in the same DSA band as the network under consideration. In addition, in [33], a default channel common to all nodes is reserved to ensure topology preservation. Their evaluation by simulation only considered 30 nodes, but a good addition was an evaluation

of a real prototype implementation of 6 nodes.

CA in WMNs using more realistic interference models is presented by Chaudhry et al. [34]. They introduce a method for building the conflict graph based on the signal-to-interference-ratio (SIR) model with shadowing for finding channel assignments in multi-radio multi-channel WMNs. The goal is to find the minimum number of non-overlapping channels required such that all incoming packets have an SIR above the required threshold for correct reception. SIR instead of SINR is used because the authors assume that co-channel interference is much larger than the noise. In this work, one radio interface of each mesh node is dedicated to control traffic only, and all radios are tuned to a common control channel. Greedy heuristic algorithms are proposed for the minimum colouring CA problem, with a worst case computational complexity of $O(L^2)$, where $L$ is the number of links. The network size considered in [34] is only 36 nodes. It is found that more realistic channel models require more frequency channels to be assigned for minimal interference.

In the extension to [34], the negative cumulative effect of individually acceptable interference levels from different interfering nodes is taken into account [35]. To do so, a conflict matrix is introduced in addition to the conflict graph. Then in the greedy heuristic colouring algorithm, the cumulative interference for each considered solution is first checked to determine whether it is within the total interference constraint specified. The solution is discarded if the constraint is not met. In another work, Chaudhry, Hafez and Chinneck consider a similar problem but use beamforming to reduce the co-channel interference [36]. While these works do address interference and SIR in a more realistic manner than elsewhere in the literature, they have a different approach from ours. The optimisation objective is to find the minimum number of frequency channels that can be used while meeting the constraints. However, none of the above three works considers the DSA scenario explicitly. They assume that all nodes have the same allowed channels. In addition, they do not consider the scenario of other users outside of the network to be optimised also causing interference. In our case, other SUs of the channels may be in a different network over which we have no control, or they can be using different standards in the same frequency bands, and so are more difficult to identify. We use the $SINR$ directly in the optimisation objective, rather than as a constraint. These works employ problem-specific heuristics rather than the metaheuristic optimisation methods we employ to address our problem. In summary, our work is a different approach to a similar, but not identical, problem.

Some works have used metaheuristic optimisation for similar and related problems to that of this work. Simulated Annealing is evaluated by Chen and Chen for CA in WMNs, while considering the interface constraint [37]. In one method of theirs, the interface constraint is modeled with a penalty function for candidate solutions. In their other method, infeasible solutions are instead converted to feasible solutions by a merge operation. Such a merge operation once again introduces the interference that the first step aimed to minimise, which is a drawback of this work. Another drawback is that the interference is considered binary, i.e., either present or not. Connectivity is ensured by assigning a channel to every link.

We now discuss some GA approaches. Sridhar et al. present a CA methodology for multi-radio WMNs that use only the Wi-Fi spectrum [38]. The optimisation goal is minimising interference. They introduce a constraint to ensure that each link is assigned a channel for topology preservation, and weight the interference objective by the link traffic, which is predicted from the previous averages. Lagrangian relaxation is used to find lower bounds. They also present a GA-based metaheuristic for solving the problem. In addition, a distributed algorithm is presented, but this requires that all radios maintain a channel assignment matrix as well as a radio usage matrix for all nodes in the network, both of which are difficult to realise. Pal and Nasipuri also present a GA, but for joint routing and channel assignment [39]. They optimise on route quality and take into account the interface constraint. A GA is employed by Ding et al. for minimising total interference and maximum link interference in WMNs with partially overlapping channels [40]. Interference from overlapping channels is also modeled as a binary factor based on a threshold. Balusu et al. combine GAs with learning automata to minimise interference in WMN CA, but for multicast tree topologies [41]. Cheng and Yang also investigated multicast tree networks [42]. They present GA, SA and Tabu search solutions for joint Quality of Service (QoS) routing and CA in multi-radio multi-channel WMNs. Subramanian et al. [43] use Tabu search to minimise binary interference, first ignoring the radio constraint and then merging channel assignments to comply with the interface constraint.

A few works use Particle Swarm Optimisation for related problems. In the most relevant work, Zhuang et al. [44] present a PSO-based CA algorithm for multi-radio multi-channel WMNs to minimise interference, again considered binary. The key difference with our work is that the same channels are allowed for use by all radios. Neighbour solutions are chosen by switching out a link on a channel to another link, rather than switching out the channel. Solutions are only considered if they are feasible assignments, satisfying the interface constraint. The fitness function is simply the total number of collisions relative to the total number of edges in the conflict graph. Ghosh et al. [45] use PSO to tackle a CA problem in mobile networks. Reassignment of channels is limited to one cell receiving a new call, and the fitness function is a linear combination of on-off states. These factors make this problem significantly simpler and less realistic than the one in our work. Abdelsalam et al. [46] investigate the use of PSO for CA in CR networks (not WMNs) by considering the mean reward and max proportional fairness objectives and considering different protection ranges for the PUs. They find that PSO generally outperforms GAs for their problem. Chakraborty et al. [47] consider PSO for the CA problem in mobile cellular networks, while

Sakamoto et al. [48] present a PSO-based algorithm for node placement in WMNs. PSO is included in the Mixed Integer Linear Programming solution to a related problem, topology control in WMNs, by Rai et al. [49]. In their work, topology control is done by scheduling with power control at the link layer, using $SINR$ as a constraint, transforming this problem into a knapsack problem, and ensuring connectivity is maintained. Unique to other works, a realistic $SINR$ model is considered instead of simple binary interference, although $SINR$ is modeled and taken into account in the problem as a constraint, which is a completely different way from our work.

Differential Evolution approaches include Da Silva Maximiano et al., who assign frequencies to base stations in Global System for Mobile Communications (GSM) using DE for minimising interference [50], [51]. Differential Evolution is also used for CA in DSA CR networks by Latif et al. [52] and Anumandla et al. [53]. In Latif's work, the objectives considered are fairness and utility, while interference with PUs and other SUs is considered only as a constraint. This problem does not have the added requirement of maintaining connectivity or topology preservation that is present in our work, because WMNs are not considered and each SU is independent. In Anumandla's work, the multi-objective optimisation encompasses three network utility functions. These are max-sum-reward, which maximises the spectrum utilisation; max-min-reward, which maximises the minimum reward of each user while satisfying the constraints on the number of channels and the required total capacity of each user; and max-proportional fairness, which is related to QoS. The researchers find that the time complexity and solution quality of DE are superior to Non-dominated Sorting GA.

Considering the existing literature, we bring novelty to this field, combining both the connectivity preservation requirement of the WMN as well as the interference avoidance requirement of DSA-using CRs. Ours is the first work to consider the near optimal CA in WMNs using soft computing methods in situations where the network uses the licensed spectrum opportunistically as SUs, with Wi-Fi as an additional option. Most other works consider Wi-Fi channels only, while a small number consider DSA CRs only. Ours is also, to the best of our knowledge, the first work to concentrate on an infrastructure DSA WMN. We approach the problem by taking into account that different nodes may have different allowed channels since the network is geographically spread out. Other works do not factor this in. We also present a new algorithm for ensuring that both the connectivity constraint and the interface constraint are met simultaneously with the constraint on which channels are allowed at each node's location. We bring to this specific problem a realistic $SINR$ model instead of an on/off interference model. The $SINR$ formulation enables extension to include adjacent channel interference. This is the first work to compare metaheuristic optimisation algorithms for such a network and scenario, with all these considerations.

## IV. PROBLEM FORMULATION

### A. NETWORK MODEL

The scenario we consider is an infrastructure wireless mesh network consisting of nodes equipped with both Wi-Fi radios and radios capable of accessing alternative spectrum, such as TVWS or CBRS, as unlicensed SUs. These WMN nodes act as APs to clients on another radio interface (this could be 2.4 GHz or 5 GHz Wi-Fi). There are also PUs of the alternative spectrum band that must be protected from interference. Hence, it is required that devices use a GLSD to get a list of channels that are allowed at their locations. This is the case for TVWS, CBRS-SAS, and Wi-Fi 6E 6 GHz. At least one node is the gateway to the Internet. To illustrate the concepts more clearly, we assume there is a single gateway node. However, in reality, there might be a gateway node from each cluster of nodes in the WMN. This allows extension to multiple gateways, as required, as the number of nodes increases. The gateway node also acts as the gateway to the GLSD. This node will gather the list of allowed channels and power levels for all nodes in the network. We also assume that the gateway node will act as a controller, gathering the average $SINR$ readings from all the nodes and performing any CA optimisation algorithm.

Ensuring that all nodes have an initial connection to the GLSD in a way that complies with regulation could be done using the method of Maliwatu [54]. In this method, nodes begin in passive scanning mode, listening for beacon frames, while one node (the gateway node, in our case) has Internet access. The node with Internet and GLSD access picks a channel and broadcasts beacon frames on this channel, along with an ordered list of alternative channels. One-hop neighbours receive this beacon frame, tune to that channel, and query the GLSD through the first node. The one-hop neighbour then selects a channel from the list of allowed channels. It can now join the network and start broadcasting beacons for the next-hop neighbour. This then allows second-hop neighbours to repeat the process and join the network through the one-hop neighbours. This process continues until reaching the outermost set of nodes.

In addition, the network may be in the presence of SU devices external to the network, which are also making use of the alternate spectrum band and so may cause interference. An example of this scenario is illustrated in Figure 1.

### B. PROBLEM STATEMENT AND MOTIVATION

Given this scenario described in Section IV-A, the question arises, "how to allocate channels to the WMN radio interfaces optimally, according to certain metrics, while ensuring compliance?". The main issues are minimising interference within the network and from external interference sources, while ensuring that connectivity is guaranteed. Connectivity must at least be maintained along the most important and critical paths (which can be determined based on the volume of traffic or other measures), and between as many nodes as possible. Different channels may be allowed for use by different nodes in the network because they are placed in

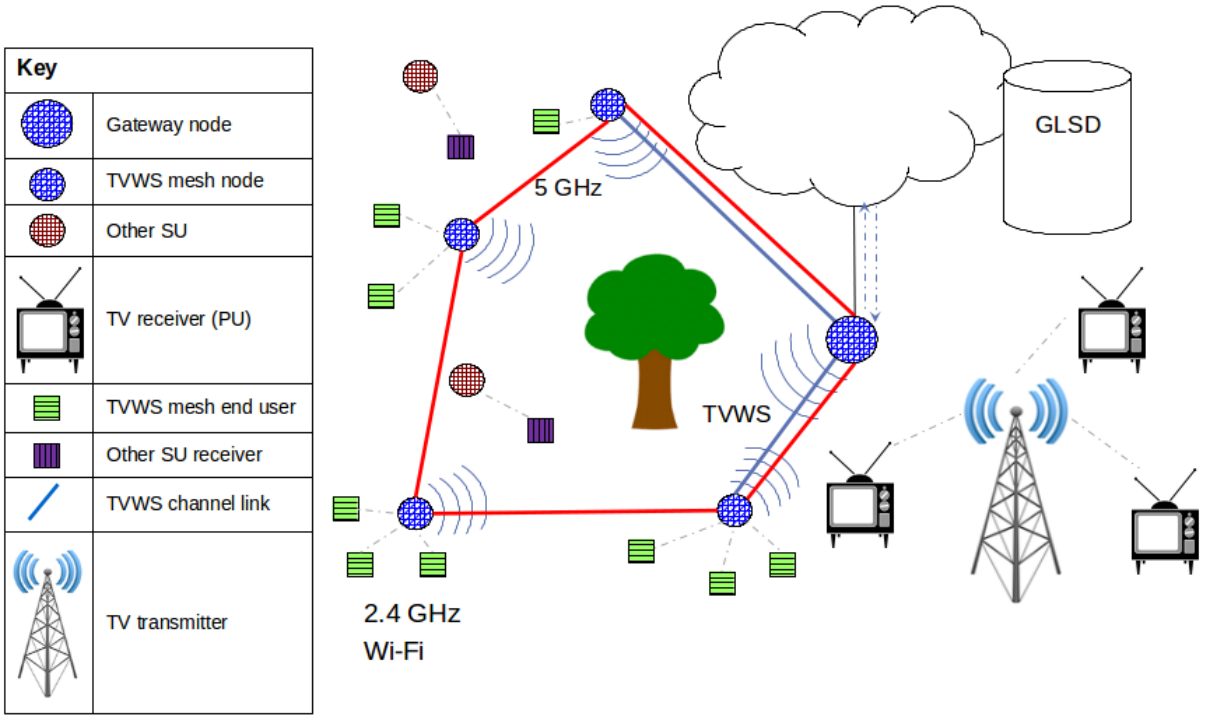



**FIGURE 1.** A triple-band infrastructure WMN using DSA. A single node is the gateway to the Internet and thus also to the GLSD. The WMN nodes have TVWS and 5 GHz Wi-Fi interfaces and can use both bands. Each WMN node acts as an AP to clients on 2.4 GHz. The DSA-WMN is in the presence of other SUs of the TVWS spectrum, which causes interference to our DSA-WMN. There are also PU TV transmitters that must be protected from interference, as well as obstructions, such as trees, that affect which spectrum bands and channels are more favourable than others.

different geographic locations. In addition, different channels may experience different levels of external interference, loss, fading, and utilisation. Hence, the problem of assigning channels optimally is an important and difficult one in this scenario.

As mentioned earlier, the CA problem is well-known to be NP-complete. In the context of a WMN, it is even more difficult and goes beyond a basic graph colouring problem. Firstly, this is because the links are not equal, as mentioned, and would require a model of a weighted graph. Secondly, this is because, while we need to avoid interference, it is also necessary to maintain connectivity and meet the interface constraint. These goals are conflicting and result in two different graph colouring problems that need to be solved at once.

We have determined that the problem is also not convex. We did so by plotting the objective function (shown by colour regions) for a scaled-down three-node (A, B, C) three-link (A-B, B-C, A-C) version of the problem, shown in Figure 2. Each of the three axes represents the channels that could be assigned to a link. The sawtooth shape in one plane for link B-C, and the presence of higher values within the low-value regions (shown by purple, magenta, and orange values inside the black region) make this problem non-convex, even in low dimensions. This justifies our use of metaheuristic optimisation algorithms and not convex optimisation algorithms.

### C. ASSUMPTIONS

The goal of the CA algorithm is to assign channels optimally to a set of links. A link is defined as a pair of radio interfaces between which traffic could potentially flow directly if tuned to the same channel. In a network, over the course of a day, the set of links used for relaying traffic will vary. The selected paths depend on the capacity of the links, which is affected by the channel allocation. On the other hand, channel allocation should consider the links used, especially those with the highest traffic load. Hence, there is an interrelationship between routing and CA. While there is an interdependence between the two problems of routing and CA, our channel assignment will be quasi-static and not change according to routing in near real-time. This is a practical and advantageous decision, rather than a limitation. Suppose the CA attempts to keep up with the rapidly changing routes, and routing is, in turn, trying to keep up with changing channel allocations. This would cause network instability, which leads to a bad user experience. Channel switching causes a loss of network connectivity during the time the Network Interface Card switches its channel and tries to re-establish connectivity, which can be on the order of seconds in reality. Optimisation algorithms, such as those we present here, are time-consuming to run and resource-intensive. This is especially true with commodity mesh radios, which are

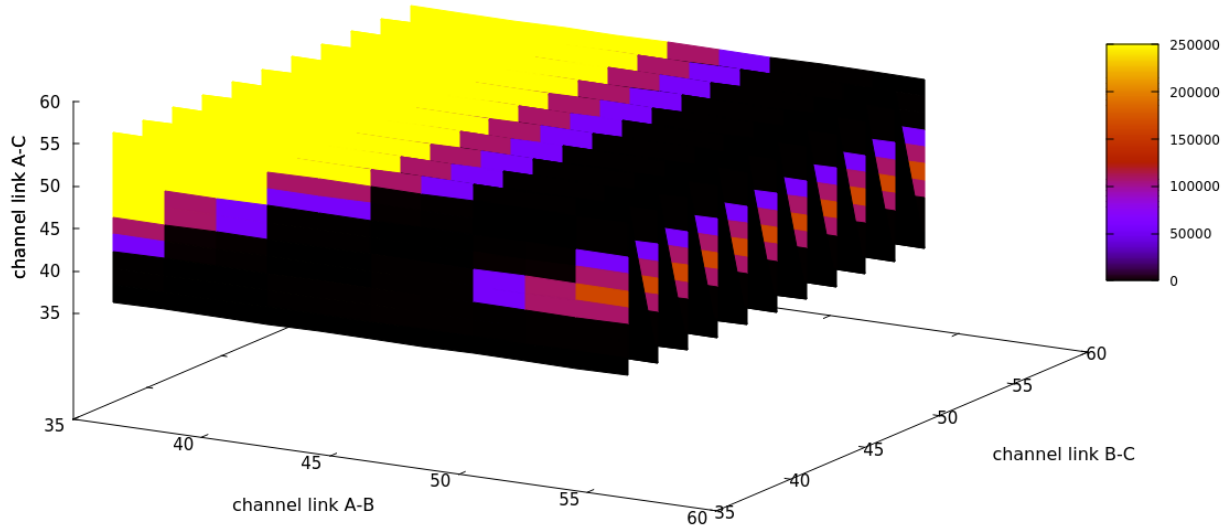



**FIGURE 2.** Map of the objective function value of the CA problem in a three-node WMN. There are three possible links (A-B, B-C, A-C), which form the three axes, and each of the three links can be assigned any of the channels. After running the WMN simulation for each possible CA, the resulting cost is plotted. The sawtooth shape in the B-C plane as well as the presence of higher values within the low-value regions in the A-B plane (shown by purple, magenta and orange patches inside the black region) show that this problem is non-convex.

resource-constrained, even if a dedicated controller node is used with more computational power than the other nodes. The constraint on computational resources means we would not want the optimisation algorithm to run often. A reasonable trade-off would be to run the optimisation once a day, for example. The optimisation algorithm could be run when the network is not busy, such as in the middle of the night. A 24-hour schedule such as this is already employed by other commercial systems for running resource management algorithms (e.g., Aruba Airmatch [55]), so it is practical and can be accepted in the industry.

Some other assumptions that apply are:

- Nodes are stationary, and the gateway node knows their locations. The mechanism by which the gateway node learns the locations is out of the scope of this work.
- PU channel use and occupancy change on a slow time scale compared to SU channel occupancy. The list of allowed channels for all nodes is known upfront before running the optimisation algorithm.
- The nodes are mostly in the same geographical area. However, some nodes on the edges may be in different geographical areas, where the GLSD defines the boundaries. If they are not, the network can be partitioned into clusters with largely overlapping allowed channel lists. For this reason, we do not present results for larger networks, as a large network could be partitioned into clusters. There are also other practical limitations on performance in the case of large networks. We consider a network of 50 or more nodes as large.
- If the nodes at the cusp of two clusters do not share a sufficient number of overlapping allowed channels in the DSA band, they can be linked by a Wi-Fi channel.
- Channel widths are fixed to the same value for all interfaces of all nodes.
- We use average $SINR$ measurements across the network in the optimisation. This is because, if the total $SINR$ over the network is large, a high throughput can be expected. $SINR$ is a direct measure of the result of changing channel assignments on the signal reception and interference experienced by nodes. These measurements will be gathered by all nodes for different possible channel assignments. An average of the samples for a particular CA will be used in the optimisation for one solution in the search space. Either the samples or the averages will be sent to the controller/gateway node to perform the optimisation. The method by which nodes obtain $SINR$ samples could be using acknowledgement (ACK) frames, similarly to Cho et al. [56].
- All of the considered links are saturated with traffic, so the average $SINR$ across the network is also a fair objective, and no other fairness criteria are necessary.

### D. MATHEMATICAL MODEL

In the usual way, we model the network as a graph $G = (V, E)$ where $V$ is the set of nodes (vertices) and edges $E$ are the links between nodes. Edges are potential links and not necessarily carrying traffic at this stage. Each edge $e \in E$ could be tuned to a particular channel at any time, i.e., $E \mapsto C$, where $C$ is the full set of considered allowed channels for the whole network (or part of the network under consideration). $C$ is the union of channels allowed in different locations of the WMN according to the GLSD. Each node $v$ has a set $C(v)$ of channels it is allowed to use. For two nodes $v_1$ and $v_2$, $C(v_1) \neq C(v_2)$ in general, although they could be equal and should have channels in common (that is, $C(v_1) \cap C(v_2) \neq \varnothing$), especially if $v_1$ and $v_2$ are neighbours. A channel is specified by a channel number, a centre frequency, and a channel bandwidth. Connectivity graph $G$ maps to a conflict graph $G_c = (V_c, E_c)$, where the vertices of the conflict graph are the edges in $G$, i.e., $V_c = E$. An edge $e' \in E_c$ exists between two vertices in $V_c$ if the two links could interfere if tuned to an overlapping channel. This could occur when the interfering signal power is above the receiver sensitivity. For example, consider two links 1 and 2 in $V_c$. An edge $e'$ exists between them if a transmission in link 2 causes power to leak into, or be transmitted on, the channel on which link 1 is operating. This can occur if the two links are tuned to the same channel. This can also happen if the links are tuned to different channels while the spectrum mask of the transmitter node is wide or the receive filtering is poor, so that power leaks into the channel on which link 1 is operating from link 2. We can model this situation as a weighted conflict graph denoted $\langle G_c(V_c, E_c), w\rangle$, where the weight $w$ represents the interference power per link. In addition, there might be sources of interference outside of the network itself, such as other transmitting SUs in the spectrum band, that can influence the reception of nodes in $G$ if they are transmitting with power in the same channel that one of the links in $E$ is tuned to. These devices are added to the conflict graph to form $\hat{G}_c$, but we note that these edges are fixed as their channels cannot be switched and their transmit power cannot be controlled.

Considering this conflict graph, we aim to minimise the conflict but maximise the wanted signal power received by each node and so maintain connectivity in $G$. We can satisfy both these requirements simply by considering $SINR$ and ensuring that all links have an allocated channel. The total $SINR$ across the network encapsulates the goals of having the highest desired received signal quality throughout the network, while also minimising conflict (interference) and noise. The optimisation objective is thus to find the channel assignment $A$, which is a mapping of $E \mapsto C$ that maximises the total $SINR$ of all nodes, given by equation (14).

$$\begin{aligned}
&\max_{A=E\mapsto C} \sum_{v\in V} \frac{P_{wanted,v}(A)}{\sum_{i\in I} P_i(A) + N} \\
&= \min_{A} \sum_{v\in V} \frac{\sum_{i\in I} P_i(A) + N}{P_{wanted,v}(A)} \\
&= \min_{A} \sum_{v\in V} \frac{\sum_{x\in V\setminus u,v} P_{x,v}(A) + N}{P_{u,v}(A)} \\
&= \min_{A} \frac{1}{\sum_{v\in V} SINR_v(A)} \\
&\equiv \min_{A} \mathbb{E}\left[\frac{1}{SINR(A)}\right]
\end{aligned} \tag{14}$$

This is subject to the radio interface constraint:

$$|A(v)| \leq R_v \quad \forall\, v \in V \tag{15}$$

and the available channel constraint:

$$A(v) \subseteq C(v)\ \forall\, v \in V \tag{16}$$

$$C \equiv \{c_1, c_2, c_3, ..., c_M\}\ \forall\, c_m \in \mathbb{N} \tag{17}$$

and a channel number $c_m$ defines a pair of centre frequency and channel width

$$c_m \mapsto (f_{c_m}, B_{c_m})$$

where:
$A(v)$ is the channel assignment of node $v$ and $|\cdot|$ indicates the size (number of channels assigned to the node);
$R_v$ is the number of radios at node $v$;
$C(v)$ is the set of channels allowed for use by node $v$;
$f_c$ is the centre frequency of channel $c$;
$B_c$ is the channel width of channel $c$;
$P_{u,v}$ is the power received at node $v$ from the transmitting node $u$;
$P_i$ is interfering power received at node $v$ from an interfering transmission $i$ over the whole channel width of channel $c$ to which node $v$ is tuned; and
$N$ is the noise power, which in Network Simulator 3 (ns3) is modeled as the product of the thermal noise ($N_t$) and the noise figure ($F_N$), as shown in equation (18).

$$N = N_t \times F_N = kTB \times F_N \tag{18}$$

where $k$ is Boltzmann's constant ($\approx 1.380649 \times 10^{-23}\ JK^{-1}$), $T$ is the temperature in Kelvin and $B$ is the channel bandwidth in Hz.

Since the $SINR_v$ is not constant and varies according to environmental conditions, we use the average of a number of samples of $SINR_v$ and so the minimisation becomes

$$\begin{aligned}
&\min_{A} \frac{1}{\sum_{v\in V} \overline{SINR_v(A)}} \\
&\equiv \min_{A} \mathbb{E}\left[\frac{1}{\overline{SINR(A)}}\right]
\end{aligned} \tag{19}$$

A transmitting node is considered interfering with $v$ if it is transmitting with received signal strength above the receiver sensitivity, and if it is in the set of nodes $V$ minus $v$ and minus the node $u$, the node transmitting the desired signal to $v$. We

only consider there to be one wanted receive signal per time slot.

We can find $P_i$ using equation (20):

$$\begin{aligned} P_i &= \int_{f_c-B_c/2}^{f_c+B_c/2} p(f_b)S_t(f_b)S_r(f_c)\,\mathrm{d}f \\ &= P_{x,v}(A) \quad \forall\, x \in V \setminus \{u,v\} \end{aligned} \tag{20}$$

where
$p(f_b)$ is the power spectral density of the interfering signal at the central frequency of channel $b$ in which the interfering node is transmitting. (It is possible that $b = c$);
$S_t(f_b)$ is the spectrum mask of the transmitter (interfering signal) centred at the central frequency of channel $b$;
$S_r(f_c)$ is the receive filter's frequency response, which is tuned to channel $c$; and all is integrated over the width $B_c$ of the considered channel $c$.

This formulation allows for extension to the case of adjacent channel interference, or interference between transmissions on any two channels, which is kept as future work.

Each transmitted signal is subject to propagation loss as well as frequency-selective fading. The received signal power at node $v$ from node $u$'s transmitted power $P_{u,v}$ (in W) (before receiver filtering) is related by the propagation loss $L$ according to the chosen propagation loss model. We apply the basic Friis transmission loss model, shown in equation (21), although the work is easily extensible to other propagation loss models, as well as real-life measured channel responses. Please note that the method is not dependent on the specific model used or on the channel response experienced. We use an isotropic antenna model in the simulation, but this can also be changed in the simulation for future work, and is also not required for our method to work.

$$P_{u,v} = P_u \frac{G_v G_u \lambda^2}{(4\pi d)^2} = \frac{P_u}{L_{u,v}} \tag{21}$$

where
$G_u$ is the transmission gain of node $u$'s antenna (unitless);
$G_v$ is the receive gain of node $v$'s antenna (unitless);
$\lambda$ is the wavelength (in m), inversely proportional to the frequency, so is affected by the channel assignment;
$d$ is the distance between the nodes (in m);
or, in dB,

$$P_{u,v}(dB) = P_u(dB) - L_{u,v}(dB) \tag{22}$$

where the path loss $L(dB)$ is the absolute value of the loss in $dB$.

Before considering interference, a link only exists if the effective received signal power on that link is above the receive sensitivity $s_v$ of the receiver node $v$. That is, the link will be pruned unless

$$\begin{aligned} P_{u,v} &\geq s_v \\ \frac{P_u}{L_{u,v}} &\geq s_v \\ SNR_v \times N &\geq s_v \\ SNR_v &\geq s_v/N \end{aligned} \tag{23}$$

$SNR$ can only be measured if it is above the receiver sensitivity/noise. This constraint reduces the number of links that require channel assignment and reduces the edges in the conflict graph that need to be considered. We also have to ensure that in the CA, condition (23) is met for critical links, so that connectivity is maintained within the network. Additionally, interference is only considered if the interference power at the receiver is above the energy detection threshold of the receiver.

In the simulation framework of ns3, frames are split into constant $SINR$ chunks, and overlapping frame chunks are considered as additional contributions to the overall noise [57]. Interfering signals are only considered as interference when the frame chunks actually overlap with those of the wanted frame at each considered receiving node in time. Preamble and payload parts of the frames are treated separately because the payload might have a higher modulation and coding rate than the BPSK-encoded preamble. Interfering signals below the energy detection threshold do not cause collisions or backoff, but are added to the interference tracker and contribute to the noise+interference in the $SINR$ calculation.

## V. METHODOLOGY

We now detail:

- the simulation setup used for our experiments in Section V-A,
- the algorithm we introduce to ensure that all solutions considered in the search space are feasible in Section V-B, and
- the implementation choices made for each of the employed optimisation metaheuristics in the remainder of Section V.

The optimisation methods generate candidate CA solutions from the solution space of possible CAs. They then obtain an average $SINR$ measurement for that CA solution, in order to optimise on average $SINR$. Over the course of a day, $SINR$ samples for some of these CAs will be taken. For those solutions for which an insufficient number of $SINR$ samples has been gathered, more samples must be gathered during the running of the optimisation algorithm. It might be necessary to generate traffic between nodes for this purpose. Since the algorithm will be scheduled to run only once a day at the least busy time, this should not cause excessive disruption to users of the network. Idle periods during the day can also be leveraged. The algorithm will start with a randomly generated feasible candidate solution and iteratively improve on that. In this work, a network simulation is used to obtain $SINR$ samples for our experimental results.

### A. SIMULATION SETUP

Simulation is an indispensable tool for network research, in particular for novel scenarios such as our own, where mesh-mode capable node hardware with DSA capabilities is not yet commercially available, or would be prohibitively

expensive to obtain for experimentation purposes. Simulation also provides a controlled environment in which to test and prove our ideas. To evaluate the performance of the algorithms, we have simulated the network using ns3. The existing ns3 simulation framework includes models for many of the network components required for our scenario, has thorough documentation and a lively support community, and is widely used. For these reasons, ns3 was our simulation tool of choice. We have built on top of the existing ns3 classes and created a module for the multi-radio multi-channel WMN simulation with interference, which models the spectrum sensing part of the DSA and the flow of traffic in a WMN.

Our ns3-dev fork contains the code used for generating our simulation results. The code can be found at [58]. We model the network scenario described in Section IV with an ns3 module called `mesh-sim`. This module consists of a configurable number of nodes arranged in a grid or random topology. The number of radio interfaces per node is configurable, but we fix the number of radio interfaces that can participate in the WMN to two for the purposes of this study. The number of radios fixed at two is a practical choice considering cost constraints in the rural and semi-urban areas that are our focus, particularly in the African context. More radios might allow an increase in throughput, but make the devices more expensive and complex, and increase the power consumption. These costs can counteract the cost benefits of using secondary spectrum bands instead of licensed bands. There is also a restriction on the number of available and utilisable channels (due to the leakages between channels, larger frequency separations might be required). Hence, it might not be possible to make use of all the radios if there are more than two.

Each interface runs the Wi-Fi MAC layer of a mesh point device. A link-to-channel mapping is passed to the `Run()` function along with a vector of links, each defined as a pair of node IDs. Channels are allocated according to the link-channel mapping specified, and traffic is generated on the specified links to saturate the links. The channels are set in such a way that they can be changed on the fly while the network is running. Channel widths are fixed at 20 MHz. We have also fixed the transmit power to the default 16 dBm, as this work does not consider optimal power levels. Optimised Link State Routing (OLSR) has been configured for the network, although our method is not dependent on the routing method, and other routing algorithms could also be applied in future. We have used the Friis propagation loss model with the frequency set appropriately, but several models are available and can be easily configured. While this propagation model is not the most accurate, it is acceptable for the purpose of analysing the performance of our CA algorithms.

Interference towards the PUs is avoided by using a GLSD to determine the allowed channels, modeled by a constrained set of channels passed to the optimisation code. We create two external SU interference sources set to interfere with our WMN node transmissions on certain channels. The waveform power is configurable, and the period and duty cycle are set appropriately to interfere with our WMN transmissions. These external interference sources also need to be avoided for our network to function optimally. Otherwise, the interference will lower the $SINR$ and the throughput in the network. To obtain the actual interference value, we use the existing `InterferenceHelper` code, where $SINR$ is calculated for every transmission. These snapshot $SINR$ values are averaged over the duration of one `mesh-sim` simulation run. In this way, we obtain the average $SINR$ in the network for a particular CA, network topology and interference configuration.

The `mesh-sim` main `Run()` function grows as $O(V + |L| + I)$ for $V$ nodes, $|L|$ links and $I$ external interference sources. Since $O(|L|) \approx O(V^2)$ (in the worst case) and $I \ll V$, the complexity reduces to $O(|L|)$. For the optimisation algorithms, each CA solution is evaluated by running the `mesh-sim` simulation for 5 s and returning the average $SINR$. A virtual 5 s was found to be sufficient, but 20 ms - 100 ms sensing interval should be sufficient in real life [59]. This is based on the fine sensing window suggested in [60]. For our population-based algorithms, we use a population size of 20 individuals. The population size of 20 individuals makes one iteration of each optimisation algorithm expensive at virtual 100 s. Therefore, we have determined that run lengths for the population-based algorithms of longer than 200 iterations are impractical, and algorithms that can find acceptable solutions in much fewer runs than this maximum are preferable.

### B. GENERATING FEASIBLE CANDIDATE SOLUTIONS

While we have used the graph analogy for this problem, it is not a simple graph colouring problem. One of the added complexities that distinguishes this problem from normal graph colouring is the interface constraint shown in equation (15). Another is that connectivity must be maintained over links, while collisions must be avoided. For all of the metaheuristic optimisation methods, we need to generate a set of possible solutions, i.e., the solution space. We can either generate each solution and then check for feasibility afterwards, or ensure feasibility within the generation procedure. Our method does the latter. We have developed a simple novel algorithm to generate candidate solutions that are feasible, instead of using a penalty function when evaluating candidate solutions. The use of a penalty function would introduce yet another problem-specific weighting parameter that would need to be quantified by experimentation, which is not desirable.

A feasible solution is one that satisfies the interface constraint and uses only the allowed channels at each node. In this algorithm, we attempt to allocate channels to all links in the network. This might not be possible. Therefore, we allocate channels to as many links as possible out of the full set. Wi-Fi channels on the Wi-Fi interface are used to ensure connectivity on the remaining links. Our algorithm is outlined in Algorithm 1.

For $|L|$ links, the worst case complexity of Algorithm 1

**Algorithm 1** Feasible channel allocation algorithm

**Input:** $C$ = allowed channels, $n_i$ = node number $i$, $L$ = set of links, $A$ = channels assigned = $\varnothing$, $r$ = number of interfaces per node
**Output:** complete $A(l)\ \forall\, l \in L$

```
for l = (n_0, n_1) ∈ L do
    if A(n_0) < r and A(n_1) < r then
        c = random channel ∈ C(n_0) ∩ C(n1)
        A(n_0) = c
        A(n_1) = c
    else if A(n_0) == r and A(n_1) < r then
        {c} = A(n_0) ⋃ C(n_1)
        if c ≠ ∅ then
            c = {c} [0]
        else
            c=choose one of C(n_0)
            A(n_1) = c
        end if
    else if A(n_0) < r and A(n_1) == r then
        {c} = A(n_1) ⋃ C(n_0)
        if {c} ≠ ∅ then
            c = {c} [0]
        else
            c=choose one of A(n_1)
            A(n_0) = c
        end if
    else
        both interfaces already assigned channels
        {c} = A(n_0) ⋃ A(n_1)
        if {c} ≠ ∅ then
            c = {c} [0]
        else
            continue
        end if
    end if
    A(l) = c
end for
∀ l unassigned, assign a Wi-Fi channel
```

is $O(|L| \cdot 2r)$ (because there are two nodes per link). If $r \ll |L|$, this can be reduced to $O(|L|)$. This is another reason to keep the number of radio interfaces low so that the computational complexity remains low. The complexity is linear in the number of links, which is exponential in the number of nodes in the network or network cluster considered, in the worst case. However, if the links considered for improved channel assignment are carefully chosen to be only the links with the highest demand, the effects of this growth could be counteracted.

### C. SIMULATED ANNEALING

A single full run of the `mesh-sim` simulation gathers a large set of sample $SINR$ values for traffic flow through a particular CA for a particular interference environment and network setup and topology. In the SA algorithm, we need the objective function (so-called "energy" value $E$) to incorporate these $SINR$ samples in a way that the desired result is the lowest cost, since SA is designed to minimise an objective function. Hence, the selected cost $E$ is based on $1/SINR$, shown in equation (24), where $j$ is the SA iteration number, $n$ is the number of $SINR$ samples per node, and $V$ is the number of nodes.

$$E_j = \frac{1}{V}\sum_{v=1}^{V}\left[\frac{1}{n}\sum_{i=1}^{n}\frac{1}{SINR_j(i)}(v)\right] = \frac{1}{V}\sum_{v=1}^{V}\frac{1}{\overline{SINR_j}}(v) \tag{24}$$

In SA, the change in cost every iteration is used to decide whether to accept or reject the particular solution. If the new solution is better than the previous solution, the new solution is always accepted. However, if the new CA has a higher cost, this worse solution is accepted with a probability given by equation (25).

$$h = exp(-\frac{\Delta E}{kT}) = exp(-\frac{E_j - E_{j-1}}{k \cdot T_j}) \tag{25}$$

where $k$ is Boltzmann's constant ($\approx 1.3806485 \times 10^{-23} JK^{-1}$) and $T_j$ is the temperature at iteration $j$.

This is realised by selecting a random value $a$ between 0 and 1 and evaluating condition (26).

$$a < h \tag{26}$$

If equation (26) holds true, the solution is accepted. If not, the solution is rejected. If equation (25) always evaluates close to 1, higher-cost solutions will almost always be accepted and the SA algorithm will take very long to converge. Conversely, if equation (25) always evaluates very close to 0, almost no "worse" solutions will be accepted and the algorithm will converge prematurely to a local minimum that may be much worse than the true optimum. Therefore, a careful balance of temperature ranges, $\Delta E$ ranges as well as $k$ must be formulated to tune the algorithm appropriately. Boltzmann's constant $k$ could be omitted from this relation (or set to 1) in practice if it makes the probability of accepting a point extremely low, leading to converging on a local minimum. Including or leaving this constant out, or even changing its value, is part of the parameter tuning required to ensure the algorithm behaves well.

The other parameter tuning that is required is the selection of the starting temperature and the temperature cooling function. A starting temperature that is too high will cause slower convergence, as will a cooling function that decreases too slowly. On the other hand, starting with too low a temperature or a cooling function that reduces too quickly may result in converging prematurely. Starting temperature and the temperature cooling function must be adjusted in consideration of the number of iterations the algorithm is expected to run for, or that is considered acceptable. We considered various cooling functions in this work, e.g., exponential multiplicative cooling and logarithmic cooling [18]. After experimentation with these different cooling strategies,

it was found that linear multiplicative cooling (equation (27)) was the most effective.

$$T_j = T_{start} - \alpha \cdot j \tag{27}$$

where $j$ is iteration count, and $\alpha$ is a constant set (to 0.02 or 0.01) by reversing equation (27) for the appropriate starting temperature (found to be 20), a final temperature of 0, and the desired number of iterations (1000 or 2000). We confirmed by experimentation that these values work well. We start with a lower temperature value of 20, selected by observation of the $\Delta E$ values for our problem, and scale the $1/\overline{SINR}$ values appropriately. With these adjustments, the algorithm is able to converge sufficiently within 1000 iterations.

The neighbour generation procedure whereby a new solution is generated is to shuffle the links and channel numbers randomly, and perform Algorithm 1. The shuffle operations together have computational complexity $O(|L| + |C|)$ since C++ `std::random_shuffle` has linear complexity in the distance between the first and last iterators of the vector to be shuffled [61]. Therefore, the neighbour generation procedure has complexity $O(|L| + |C|) + O(|L|) \approx O(|L| + |C|)$. Finding the cost of every solution has the complexity of the `mesh-sim Run()` function, which is also $O(|L|)$. Therefore, the overall complexity of our SA implementation grows as $O(|L|)$ if we assume $r \ll |L|$ and $|C| \ll |L|$. If we take into account the number of iterations, the complexity is $O(|L|N)$ for $N$ iterations. Being linear in the number of links, this complexity is much lower than finding an exact solution would be. To find an exact solution by brute force would require trying all of $|L|^{|C|}$ possible solutions.

### D. GENETIC ALGORITHM

For the GA, we need to define the selection method, population size, number of generations, and the mutation rate. We encode a genome as a link→channel mapping, where the links are all node pairs possible in the WMN and where condition (23) is met. To generate a new genome, we randomly shuffle the set of links, randomly shuffle the set of allowed channels, and use Algorithm 1 to generate a feasible genome. We then generate a population by generating a number of genomes. Each instance of C++ `std::random_shuffle` has linear complexity in the distance between the initial and final iterators minus one [61]. This results in complexity $O(|L| + |C|)$ for generating a genome and $O(P(|L| + |C|))$ for generating the initial population of size $P$. We determined from experimentation that $P = 20$ functions well without excessive computational burden. This population size is confirmed as a good choice by Kononova et al. [62], who find that a population size of 20 presents less structural bias than populations of 5 or 100 individuals in general.

Both Roulette Wheel selection and Linear Rank selection methods were implemented. For the Roulette Wheel selection, we generated a piecewise constant probability distribution, where the intervals are 1 + the population size and the weights are the fitness values of the individuals in the population. For Linear Rank selection, we sort the chromosomes by their inverse fitness values so that the genome with the highest fitness has the lowest rank (highest number). We then create a piecewise constant probability distribution of the ranks and select two parent chromosomes randomly according to that distribution. We use the C++ `std::sort` algorithm for Linear Rank selection, which has complexity $O(P \log_2(P))$, where $P$ is the size of the population [63]. It was found that Linear Rank selection outperforms Roulette Wheel selection, so only the results for Linear Rank selection are presented in this paper. We select as many parents as the current population, and each pair of parents generates two children. The previous generation is eliminated once they reproduce, so the size of the population remains stable.

Once two parents have been selected, the next operator is crossover. Single-point crossover is used. In this implementation, the crossover operator randomly selects an index in the genome (a link) greater than the first and smaller than the last as the crossover point. We then split both parents at this crossover point and generate two new children by joining the first section of the first parent with the second section of the second parent, and the first section of the second parent with the second section of the first parent. This crossover is illustrated in Figure 3, with the two selected parent chromosomes at the top and the generated offspring at the bottom. The single-point crossover operation includes two calls to C++ `std::find`, which has complexity up to $O(|L|-2) \approx O(|L|)$ [64], and four calls to `std::map::insert`, which has worst case complexity of $\approx O(|L| \log(|L|))$ [65] for our variables.

Experiments are run for mutation rate parameter settings of both 0.5 and 0.25. The mutation probability is implemented by choosing a random number in $(0, 1)$; then, if the random number is less than the mutation rate (0.5 or 0.25, in our case), mutation is performed. The mutation operation is done by randomly selecting one link and randomly selecting a new channel for that link, and replacing the currently assigned channel with the new one. This operation has logarithmic complexity in the size of the genome, i.e., logarithmic in $|L|$. The 0.5 probability was found to provide a suitable trade-off between exploration and exploitation for the relatively small population size, and the specific problem. This follows the findings of [66], who find a good region of performance between mutation rates of 0.4 and 0.6, although this was for a small population of 6 individuals. Deb and Agrawal [67] also found that smaller population sizes (less than 100 individuals) require higher mutation rates. These researchers found superior performance for a combination of the population size of 20 individuals, the mutation rate of 0.5, and a high crossover rate of 0.9 to lower mutation rates. We also ran experiments with the lower mutation rate of 0.25 to determine whether convergence can be achieved in fewer iterations. Lower mutation rates can be considered more traditional [68]. Several researchers have found that a lower mutation rate is more optimal, even for small and medium-sized populations of 4-20 individuals [69]–[71].

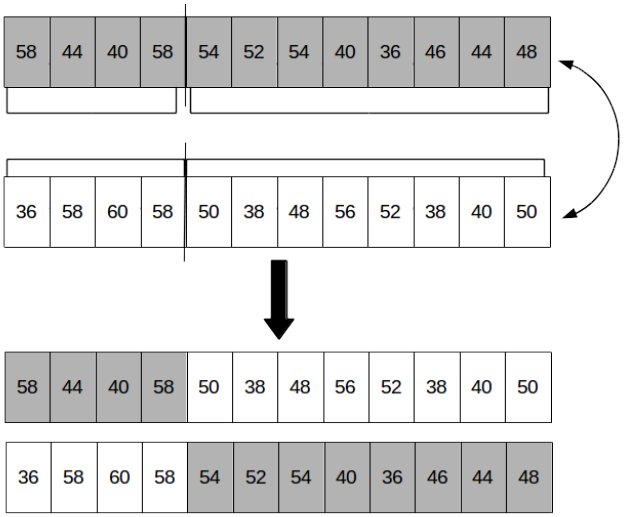



**FIGURE 3.** Crossover of channel allocations in Genetic Algorithm. The two parent chromosomes are the top two CAs, and they produce the two offspring shown below the arrow, by mixing the first section of the first parent with the second section of the second parent, and the second section of the first parent with the first section of the second parent chromosome.

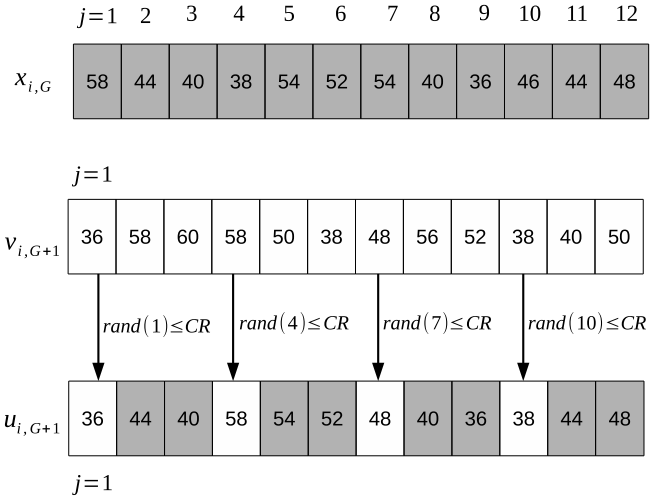



**FIGURE 4.** Crossover of channel allocations in Differential Evolution. $rand(j) \leq CR$ refers to selecting a random number in the interval $(0, 1)$ and checking whether it is less than the crossover rate. If it is, then the value of the mutant is selected for crossover with the target vector $x_{i,G}$.

Overall, the computational complexity of our implementation of GA is

$$\begin{aligned} &O(P(|L|+|C|)) + O(G \cdot (P|L| + P + P\log_2(P) \\ &\quad + P(|L| + |L|\log(|L|) + \log(|L|)))) \\ &\approx O((P(|L|+|C|)) + O(GP(|L| + \log_2(P) \\ &\quad + |L| + |L|\log(|L|) + \log(|L|)) \\ &\approx O(GP(\log_2(P) + |L|\log(|L|))) \end{aligned} \tag{28}$$

assuming $|C| \ll |L|$ and $r \ll |L|$. Each generation of GA contains a population of size $P$ and all evaluations done on an individual genome is the equivalent of one iteration of SA. We denote this iteration equivalent as "function evaluations". This is the part inside $O(GP(\cdot))$ in Equation (28). Here we see that per function evaluation, GA has complexity $O(\log_2(P) + |L|\log(|L|))$. This is in contrast to our SA implementation where each function evaluation (iteration) has complexity $O(|L|)$. Hence, our implementation of GA is significantly more complex than SA.

### *E. DIFFERENTIAL EVOLUTION*

In DE, individuals in the population are called *agents*. We generate agents $x_{i,G}$ by producing link-to-channel-index mappings, which represent CAs, and we generate a population $x_G$ by pushing agents to a population vector. Each link is a dimension of the agent vector. For each generated mutant vector $v_{ij,G+1}$ $\forall$ $j = 1, ..., D$, a channel value is calculated with equation (1). This calculation may cause the computed channel value to be outside of the allowed channels and no longer an integer value. Therefore, the calculated channel value must be moved back into the allowed channel list. This is done by having the channel indexes wrapped around, and the result is rounded off to the nearest integer. For example, if there are 13 possible channels and the mutant in that dimension is calculated to be at -8.3, the resulting channel index will be $\lceil -8.3 \rceil + 13 = 5$ (for indexes starting at 1). If, instead, the mutant is calculated to be at channel 14.8 in that dimension, the resulting channel index will be $15 - 13 = 2$.

The crossover operation we use for DE is illustrated in Figure 4. Here we see that the position of $x_{i,G}$ is crossed with that of $v_{i,G+1}$ on links 1, 4, 7 and 10 where the random number generated (rand$(j)$) was $\leq$ to the crossover rate $CR$ to form the trial vector $u_{i,G+1}$. This step has complexity $O(|L|)$.

Selection is implemented as per equation (3). The selection in DE requires an extra evaluation of the fitness function for the trial vector compared to GA or PSO. This results in a complexity of $O(P \cdot 2|L|r)$, reducing to $O(P|L|)$. Even though DE reduces to the same complexity per function evaluation as SA ($O(|L|)$), we must note that it will take longer because of the extra calculation of the fitness function for every function evaluation. On the other hand, it is less complex than our implementation of GA.

Georgioudakis and Plevris [72] mention that, in DE, the parameter values are very problem-specific and the results are sensitive to the values of $F$ and $CR$. We, in fact, found good performance with a range of parameter values. Good performance was observed with $DE/rand/1/bin$ variation, so we do not present results for other variations. Following the recommendations of Storn and Price [20], we start with a crossover ratio of 0.9 and set $F$ to 0.9, as was one of the combinations of parameters in the study of Storn and Price. Parameter settings of $F = 0.4$ and $CR = 0.5$ were also used with similarly promising results. We also tried combinations of $F = 0.8$ and $CR = 0.9$, $F = 0.9$ and $CR = 0.1$, $F = 0.5$

and $CR = 0.9$, $F = 0.6$ and $CR = 0.9$, and $F = 0.6$ and $CR = 0.5$. All these parameter settings showed good results, as is shown in Section VI.

### F. PARTICLE SWARM OPTIMISATION

We define some quantities for the PSO implementation, and the way in which the method has been converted to a discrete algorithm. In our PSO applied to the CA problem, the position of a particle, $\boldsymbol{x_i}$, is a link-to-channel-index mapping. It is equivalent to a channel assignment ($A$), i.e., $\boldsymbol{x_i} \equiv A$, but using the channel index instead of the channel number. Each link is a dimension of the position of the particle (so a CA with 120 links will have a dimension of 120). It is important to use the channel index and not the channel number for velocity update operations, so that a velocity or displacement can have a consistent meaning. Moving with a displacement of +2 should be going up two available channels, regardless of the actual channel number or gaps between the defined allowed channels. There are often inconsistent gaps between channel numbers, e.g., if a channel number is 116, then the next available channel is 120, but from channel number 144, the next available channel number could be 149. This would mean that the velocity (displacement) cannot have a consistent meaning. If we use the channel index, this inconsistency is corrected. For the same reason, we cannot use the continuous frequency space instead.

To generate a swarm, we generate a set of $P$ particles (random positions in the solution space), and calculate their fitness. This initialization has complexity of $O(P(|L|+|C|))$. We have already shown the original canonical PSO velocity calculation [22] in equation (8). We use this equation in its discrete form for our implementation, by rounding the values to the nearest integer, as shown in equation (29).

$$\begin{aligned}\boldsymbol{v_i}(t+1) = \text{round}\,[\,&\omega \boldsymbol{v_i}(t) + c_1 r_1(t) \times (\boldsymbol{y_i}(t) - \boldsymbol{x_i}(t)) \\ &+ c_2 r_1(t) \times (\hat{\boldsymbol{y}}(t) - \boldsymbol{x_i}(t))\,]\end{aligned} \tag{29}$$

When updating the position, we again make the channel indexes wrap around, so if the velocity moves the particle to a position outside of the bounds of the number of channels, it starts counting back from the beginning. For example, if there are 13 possible channels and the particle link is currently on channel 4, and the velocity moves the particle in that dimension -8 channels, the resulting channel index will be $4 - 8 + 13 = 9$ (for indexes starting at 1). If, instead, the velocity is 11, the resulting channel index will be $4 + 11 - 13 = 2$. The position and velocity update stage per particle has complexity $O(|L|)$.

A note in Bratton and Kennedy's work that no swarm size between 20 and 100 proved significantly inferior or superior to the others also informed our choice of a swarm size of 20 particles. In addition, Kononova et al. [62] show that a PSO with a population size of 20 exhibits satisfactory performance in terms of structural bias, while population sizes of both 5 and 100 display more structural bias. If a choice of 20 particles does not perform inferior to a larger swarm size and if there is no other benefit to a larger swarm, we would elect to save on computation time and choose a population size of 20. We confirmed our choice with experiments.

For the PSO velocity update parameters using the standard PSO, it has been determined that $|\omega| < 1$ [73] or $0 \leq \omega < 1$ [74] is required to ensure convergence. We started with the recommended values [75] for $\omega$, $c_1$ and $c_2$, where $\omega = 0.72984$ and $c_1 = c_2 = 2.05$ (so they add up to 4.1). With these values, convergence was not observed after 1000 iterations for most attempted runs, although there was one run that appeared to converge within 1000 runs. We believe that this value of $\omega$ combined with $c_1$ and $c_2$ values over 2.0 results in the inertia being dominated by the social and cognitive components, so that if the initial values found are bad, there is more likelihood of moving around in a bad neighbourhood. However, if the initial values are good, then the particles move towards these good values, which is why one run happened to perform fairly well. We then increased $\omega$ to 1.05, following the recommendation of [23], and observed a slight improvement. This larger $\omega$ encourages more exploration of the search space, which is advantageous at the beginning of a run of PSO. However, we did not observe strong convergence because this value is not within the convergence region. Strangely, for Variant 1, using a large value of $\omega = 1.5$ was quite successful in causing convergence for a 9-node experiment, but not, in general, for larger networks. In their discussion on optimal inertia weight, Shi and Eberhart propose an adaptive $\omega$, starting at 0.9 and ending at 0.4 [23]. In general, an adaptive inertia weight follows equation (30) for finding the weight at each iteration. We found that this version performed slightly better overall than any single value of $\omega$ that was tried.

$$\begin{aligned}\omega(t) &= \omega_{final} + \frac{t_{max} - t}{t_{max}} \times (\omega_{initial} - \omega_{final}) \\ &= \omega(t_{max}) + \frac{t_{max} - t}{t_{max}} \times (\omega(t_0) - \omega(t_{max}))\end{aligned} \tag{30}$$

Wei et al. introduce an elite PSO with mutation [76]. In this method, elite and bad particles are distinguished after some iterations. Bad particles are replaced by the same number of elite particles. To prevent the loss of diversity caused by this replication of particles, mutation is then applied to the elite particles before using them to replace the bad particles. We have developed our own form of this method, called "*PSO with bad replacement*". In our method, particles are monitored in comparison with the rest of the swarm and are labelled bad if, after $t_{bad}$ iterations, they are $bad\%$ worse than the global average. We have set $t_{bad}$ iterations to 5 and $bad\%$ to 5000%, or 50 times the swarm average, by observation of the magnitude of the bad cost values. These particles are replaced with an equal number of new randomly generated particles. Our method introduces diversity and prevents bad particles from ruining the swarm in one simple step, unlike Wei et al.'s method that requires two steps. The "bad replacement" step adds a factor of $|C|$ to the complexity

whenever a particle is replaced, but this is not significant since, in general, $|C| \ll |L|$, and replacement occurs rarely.

The complexity of our implementation of PSO is $O(P(|L| + |C|)) \approx O(P|L|)$. This is similar to DE, but without the extra factor of 2 that is present in DE because the fitness of trial vectors also has to be determined. PSO is lower in complexity than GA. By comparing the number of function evaluations instead of only considering the number of iterations, PSO and SA have similar computational complexities.

## VI. RESULTS

Simulations were run on a Dell Latitude with 7.7 GiB of memory, an Intel Core i5 processor at 4 × 2.4 GHz cores; as well as a T2 large Amazon Web Services EC2 instance with 8 GiB of memory and 2 virtual CPUs, both with Ubuntu 16.04 Operating System, and using ns3-dev version [58] forked from the main ns3 GitHub.

In each iteration of the optimisation algorithms, the WMN simulation is run for a period of 5 s (virtual). From the WMN simulation we gather samples to calculate the value of equation (24). The 5 s interval was found to yield sufficient $SINR$ samples for the average to be meaningful. In the `mesh-sim` simulation, nodes are set up in an equally spaced grid, or in random positions within a disc according to a uniform distribution for the polar coordinates. Each node has two interfaces (representing the DSA band interfaces). Constant bitrate UDP traffic is generated at the transmit node for every link in the network so as to saturate the links. Packets will be received on the other side if there is a common channel between the two nodes and the received signal is above the receiver sensitivity. The interference is included in the $SINR$ measurement using ns3's `InterferenceHelper` class, and interference is counted only if the overlapping packet chunk is above the sensitivity of the receiver. The simulation parameters are given in Table 1.

**TABLE 1.** Parameters used in simulations

| Parameter | Value |
|---|---|
| Network size | 9-49 nodes |
| Number of interfaces | 2 |
| Number of intersecting channels | 13 |
| Distance between grid nodes | 100 m (vertical and horizontal) |
| Max radius of random disc | 350 m |
| Channel bandwidth | 10 MHz |
| Propagation loss model | Friis |
| Propagation delay model | ConstantSpeed |
| Packet interval | 0.01 s |
| Packet size | 1024 bytes |
| Interferer waveform power | 0.2 W |
| Interferer centre frequencies | 498 MHz and 522 MHz |
| Interferer waveform period | 0.0007 s |
| Interferer waveform duty cycle | 1 |
| Error rate model | NistErrorRateModel |
| Mesh routing algorithm | OLSR |

The use of 13 channels was set as a worst-case scenario for computational load in terms of the size of the search space. If the number of overlapping allowed channels is less than this, the number of options is smaller and fewer iterations are required for convergence.

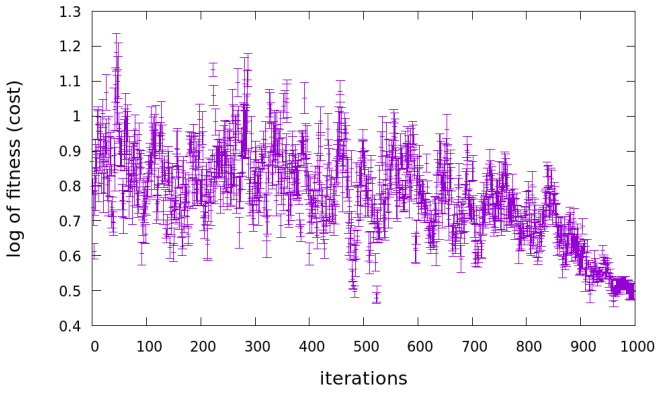
### A. SIMULATED ANNEALING

The means of the natural logarithm (ln) of the costs (scaled value of equation (24)) obtained from 10 different runs of SA at each iteration over time are illustrated in Figure 5, along with the standard error, shown by the error bars, across the different runs. We have used the natural logarithm of the cost values obtained for better visibility, since the data has a large range.

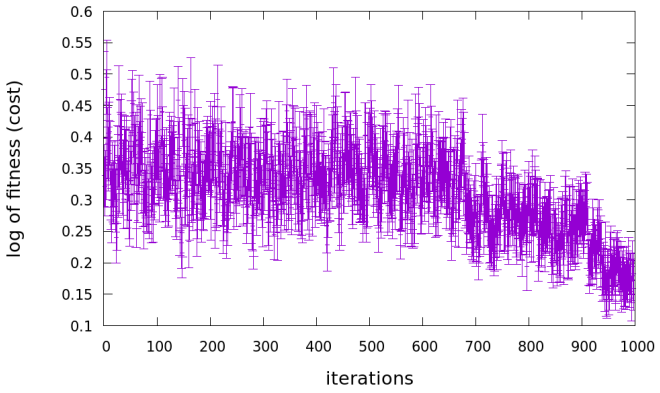
We can see in each case that the solutions found had a large variance towards the beginning of the runs, but different runs slowly converge to similar solutions with smaller variance (and standard error) as the number of iterations increases. Acceptable levels of convergence are observed within 1000 iterations, although we will see in Table 6 that much better results are obtained after 2000 iterations. We also observe in Figure 5 that the plots do have a general trend downward, but this trend is less visible because the error bars are still relatively large. The convergence is the clearest for the 9-node network, ending with a small standard error. Then, for 16 nodes, the convergence is somewhat less clear, ending with a larger standard error, and for 49 nodes, there is still a large standard error between the results of the different runs by the end of 1000 iterations. We can deduce that SA does not scale well since the convergence deteriorates as the network size increases.

### B. GENETIC ALGORITHM

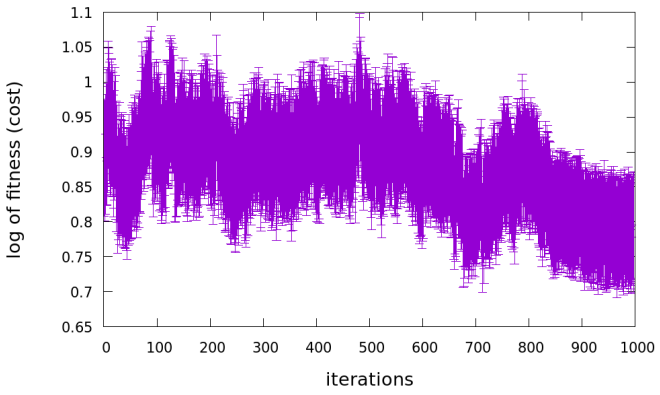
For each run of the GA, we find the mean cost (scaled value of equation (24)) of the population at each iteration and then plot the natural logarithm of the mean over the 10 different GA runs of this mean population cost per iteration. These results are shown for different network sizes in Figures 6 and 7. The population size was 20 for all runs, and the mutation rates were 0.5 (Figure 6) and 0.25 (Figure 7), respectively. For a population size of 20, 20 function evaluations are required per iteration. We observed convergence within 50 iterations in most instances, although a longer run of 500 generations ensures better results. Interestingly, we observe that the convergence shape of the results improves as the network size increases, even though the final best results deteriorate slightly with increasing network size. The 49-node network has a very clear convergence shape, looking similar to an exponentially decreasing function. For the 16-node network, it is still visible but less clear. For the 9-node network, this shape is less clear, and there is still a significant variation in the population averages towards the end of the 100 iterations. This is despite there being little variance between the best values at the end of 100 iterations, as we see in Table 2. It is likely that in a larger network there are more opportunities for rerouting traffic away from links that experience high interference. On the other hand, in a smaller network, there are fewer options for rerouting, and so interference has not been avoided as well by all individuals (CA solutions) in the




(a) 9 nodes

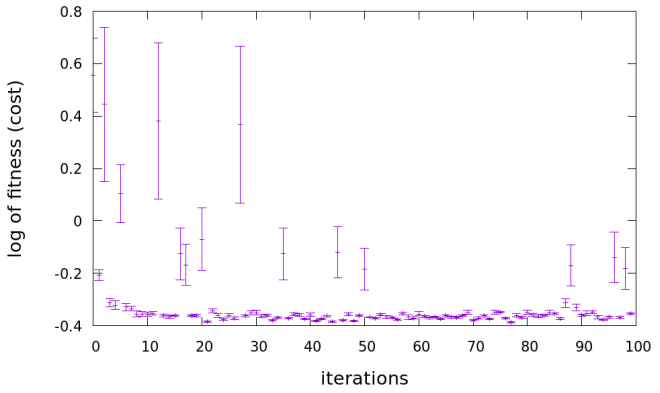



(b) 16 nodes

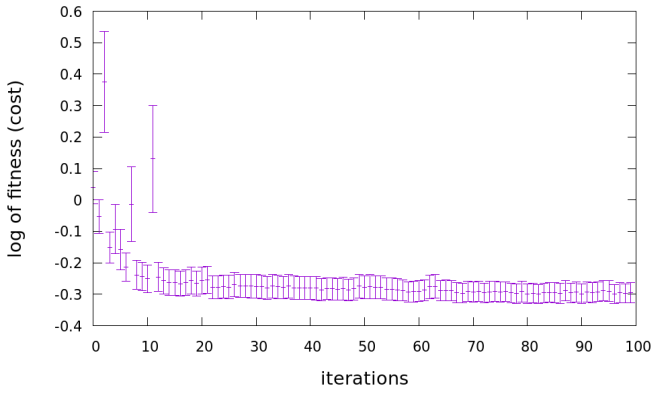



(c) 49 nodes

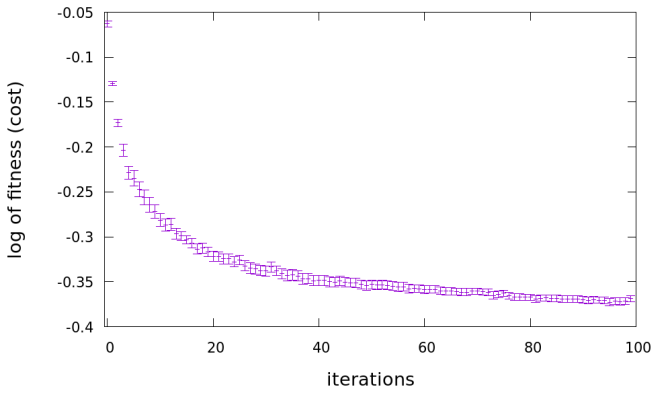
**FIGURE 5.** Mean and standard error (error bars) of the natural log (ln) of the cost obtained from 10 runs of SA at each iteration or function evaluation over the running time (iterations)




(a) 9 nodes mutation rate = 0.5




(b) 16 nodes mutation rate = 0.5

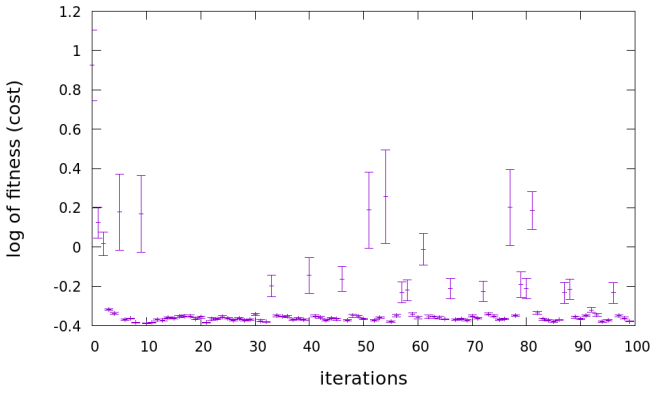



(c) 49 nodes mutation rate = 0.5

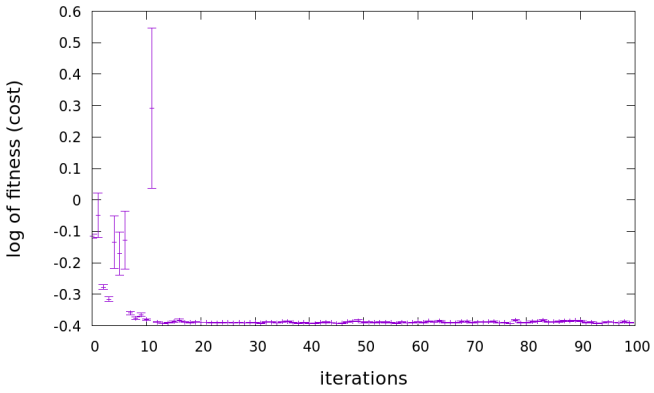
**FIGURE 6.** Mean and standard error of the natural logarithm (ln) of the mean costs of populations of 10 runs of GA over the running time (iterations) using a mutation rate of 0.5. The mean cost per iteration is the average of the costs of 20 function evaluations (the 20 individuals in the population).

population. However, the best individual in a population - and indeed several other individuals - are able to find CA solutions that avoid interference successfully despite fewer routing opportunities, as seen in Table 2.

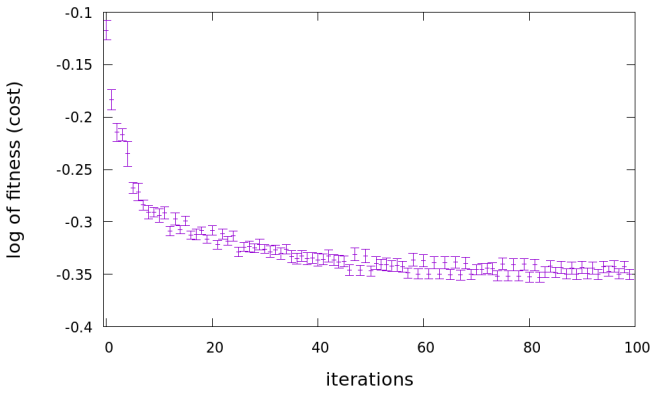
In Table 2, we show the mean over the 10 runs of GA of the final solution found, i.e., the cost of the best individual in each population at the end of 100 iterations along with the standard deviation ($SD$). We see that for 9 nodes and a mutation rate of 0.5, the same solution was found by all sample runs, so we assume this is the actual minimum for the problem. The solutions for a 16-node network have a slightly higher standard deviation, so we cannot be sure to have found the actual minimum within 100 iterations, even though the solution is still very good. The results for 49 nodes after 100 iterations have a slightly larger standard




(a) 9 nodes mutation rate = 0.25




(b) 16 nodes mutation rate = 0.25




(c) 49 nodes mutation rate = 0.25

**FIGURE 7.** Mean and standard error of the natural log (ln) of the mean population cost of 10 runs of GA over the running time (iterations) using a mutation rate of 0.25. The mean cost per iteration is the average of the costs of 20 function evaluations (the 20 individuals in the population).

deviation still, although this value is still considerably better than some of the initial solutions found and much better than the worst solutions observed. Surprisingly, there was slightly more variation in the results using a mutation rate of 0.25. It is possible that the lower mutation rate results in there not being enough diversity in the populations and thus premature convergence. It is less likely to be owing just to statistical variation because the higher standard deviation is observed for all network sizes.

**TABLE 2.** Comparison of the final results of GA (scaled cost or value of equation (24)) of the best individual at max iterations of 100)

| | 9 nodes | 16 nodes | 49 nodes |
|---|---|---|---|
| Mutation rate | mean ($\pm SD$) | mean ($\pm SD$) | mean ($\pm SD$) |
| 0.5 | 0.4 ($\pm$0.0) | 0.4 ($\pm$0.0004) | 0.4 ($\pm$0.02) |
| 0.25 | 0.4 ($\pm$0.0005) | 0.4 ($\pm$0.0005) | 0.4 ($\pm$0.04) |

### C. DIFFERENTIAL EVOLUTION

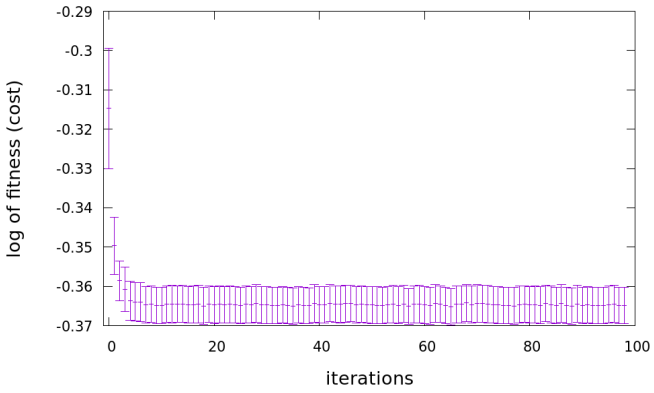
**TABLE 3.** Comparison of the final results of DE (scaled cost or value of equation (24)) of the best individual at max iterations of 100) with parameters $F = 0.9, C = 0.9$

| | 9 nodes | 16 nodes | 49 nodes |
|---|---|---|---|
| $F, C$ | mean ($\pm SD$) | mean ($\pm SD$) | mean ($\pm SD$) |
| 0.9, 0.9 | 0.4 ($\pm$0.0006) | 0.4 ($\pm$0.0006) | 0.4 ($\pm$0.0007) |

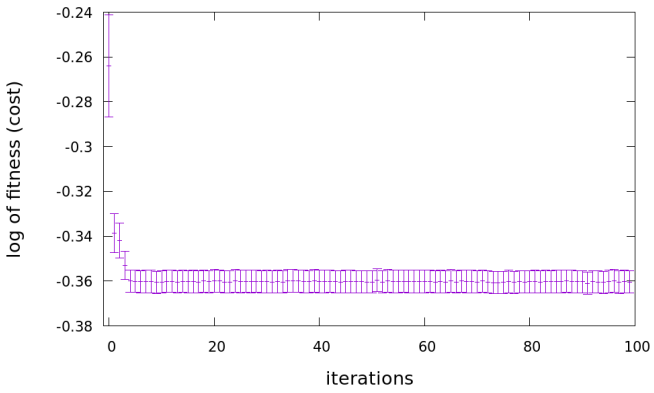
For DE, the population size was also fixed at 20 for all experiments. Initial experimentation showed good results with $F = 0.4, CR = 0.5$, $F = 0.9, CR = 0.1$, $F = 0.5, CR = 0.9$, and $F = 0.9, CR = 0.9$, as seen in Figure 8 and comparing with Figure 9. Since there was no significant difference in the results for these parameter values, we continue to present only the log results for $F = 0.9, CR = 0.9$. Table 3 shows the final results of DE (scaled cost of the best individual at max iterations of 100). Figure 9 shows the mean of the natural logarithm of the mean cost of the populations over 10 runs of DE at each iteration. The error bars indicate the standard error of these log means. We captured the results for runs of 500-1000 iterations. However, there is no significant change from 100 iterations, and so Figures 8 and 9 show only up to 100 iterations. In fact, there is no significant change after fewer than 50 iterations, as we can see from those figures. DE converges very quickly and produces good results. In practice, DE could be run for no more than 50 iterations to obtain good channel assignments. It could even be run for only 20 iterations and still produce satisfactory results.

### D. PARTICLE SWARM OPTIMISATION

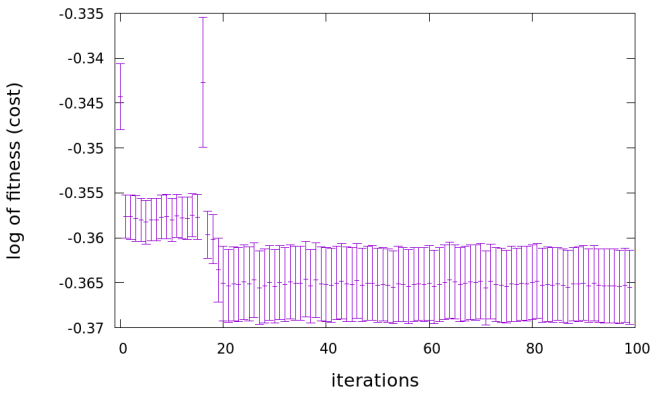
For PSO, we attempted various parameter combinations with the swarm size fixed at 20 individuals. Many of the attempted parameter combinations were unsuccessful in showing a clear convergence pattern within 1000 iterations. For a successful run, we expect to see an initial exploratory phase, where bad solutions might be observed, followed by an observable reduction over time of the average cost of particles in a swarm along with a reduced variance in the cost values across particles as the algorithm convergences towards a lower value over time. A combination of parameters that results in runs with this pattern was difficult to find. For




(a) 9 nodes $F = 0.4, C = 0.5$

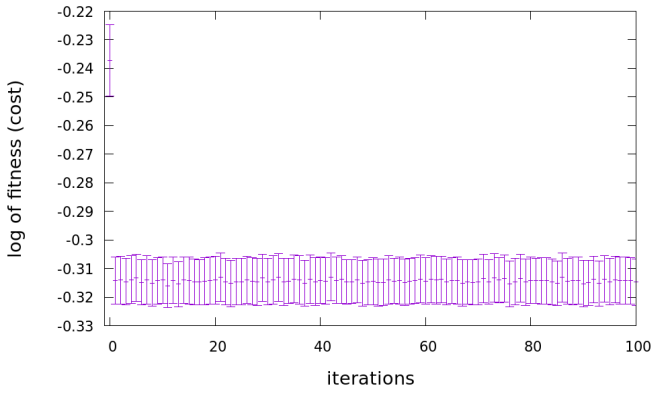



(b) 9 nodes $F = 0.9, C = 0.1$

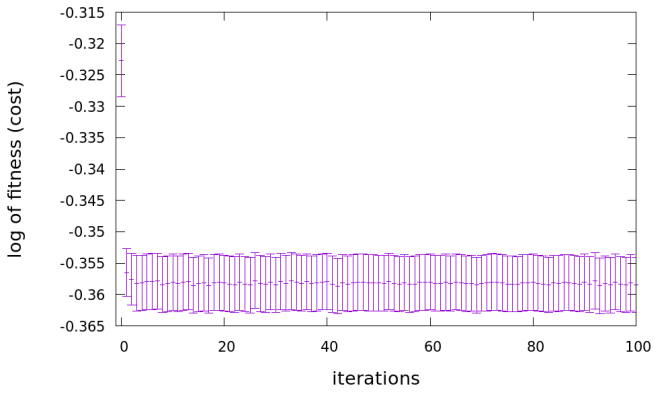



(c) 9 nodes $F = 0.5, C = 0.9$

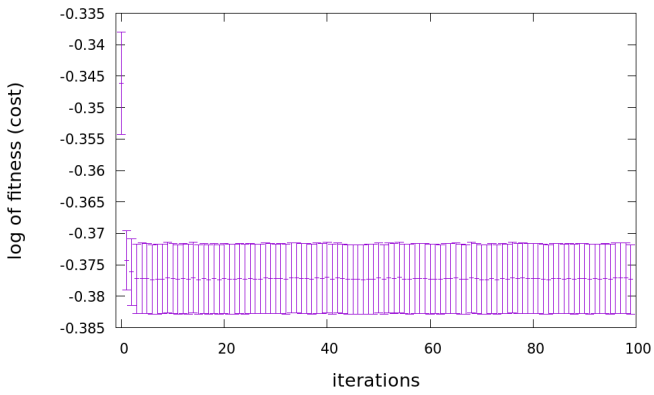
**FIGURE 8.** Mean and standard error of the natural log $(\ln)$ of the mean costs of populations of 10 runs of DE over the running time (iterations) for a 9-node WMN. 1 iteration=20 function evaluations.




(a) 9 nodes $F = 0.9, C = 0.9$




(b) 16 nodes $F = 0.9, C = 0.9$




(c) 49 nodes $F = 0.9, C = 0.9$

**FIGURE 9.** Mean and standard error of the natural logarithm $(\ln)$ of the mean population costs for 10 runs of DE. 1 iteration=20 function evaluations.

example, in Figure 10, we used the recommended values of $\omega = 0.72984$ and $c_1 = c_2 = 2.05$ [75] using velocity update method of variant 1. There is no convergence of the mean cost of the swarm within 500 iterations. We also plot the values for a much longer run of 100000 iterations on a small 3-node network in Figure 11, which shows a clear convergence towards smaller values over the many iterations of this long run. Hence, we can be certain that the lack of convergence observed within 500 iterations is not caused by implementation errors, but that PSO requires a longer run to show sufficient convergence for this problem. While it may appear that Figure 11 has no values from 90 000 iterations on, this is just because the values are too small compared to the large scale to be observed.

We found more promising results using an adaptive $\omega$ as per equation (30). In the next figures, we show the mean

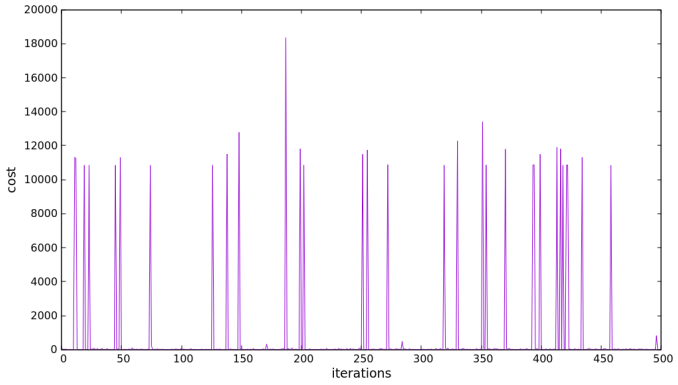



**FIGURE 10.** Unsuccessful run of PSO with 9 nodes variant 1, with $\omega = 0.72984$ and $c_1 = c_2 = 2.05$. We note that with the recommended parameter values there is no clear convergence within 500 iterations (where 1 iteration=20 function evaluations). These parameter values perform poorly.

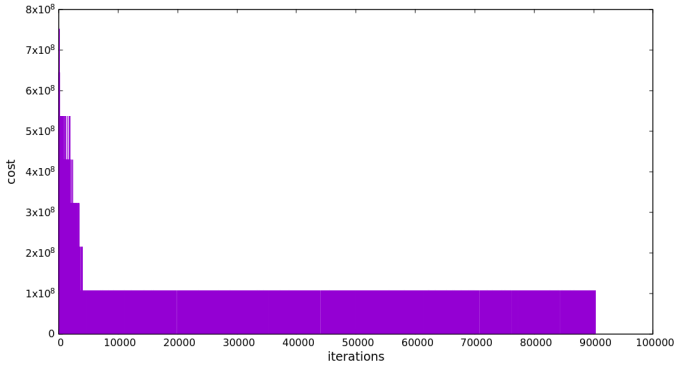



**FIGURE 11.** Costs over long run for 3-node WMN, where 1 iteration=20 function evaluations.

per iteration of the natural logarithm of the mean cost of each swarm over 10 runs of PSO. The error bars indicate the standard error.

In the 9-node network (please refer to Figure 12), we can see some convergence within 500 iterations using variant 1 (equation (8)) and using variant 5 (equation (12)); there is even convergence within 100 iterations for variant 5. For variant 6 (FIPS), it is unclear, but there is some convergence within 500 iterations.

The behaviour for larger 16-node and 49-node networks is different, as seen in Figures 13 and 14. We show up to 500 iterations for clarity of the figures, but this behaviour continues up to 1000 iterations. We can see that there is no clear convergence pattern within the iterations shown. However, the values around which the fluctuations occur are small, and the fluctuations are small if one considers the scale of the y-axis. This would indicate premature convergence has occurred in the cases illustrated in Figure 13 and Figure 14. This is a common issue experienced with PSO [77], [78]. We observe somewhat more exploration of the search space for variant 6 than for variants 1 and 5, especially in the case of 16 nodes. We can also observe that in the cases of Figures 14, the first few values were lower than the final value settled on. This also indicates premature convergence.

The runs using variants 2, 3, and 4 all failed to converge or were no better than variants 1, 5, or 6, so we did not consider these variations further nor present these results. We observed poor results when using the recommended values of $\omega = 0.72984$ and $c_1 = c_2 = 2.05$ and better results using adaptive $\omega$. However, while we did not observe a clear pattern of convergence for the mean cost values of all particles over 10 different runs in Figures 12, 13, and 14, the final results of PSO were good. The final result is the cost value of the best particle at the end of the run. These values are presented in Tables 4 and 5. It is noteworthy that good results were observed even for the larger network, although the standard deviation of the final values increases as the network size increases. Also important to note is that there is very little difference in the solutions obtained after 100 iterations and 500 iterations, as we can see by comparison of Tables 4 and 5, except for the smallest network. Probably PSO takes longer to converge for the larger networks, but such tight convergence as observed for 9 nodes in Table 4 is not required. This means that there is no materially significant advantage to running the algorithm for longer than 100 iterations.

**TABLE 4.** Comparison of the final results (scaled value of equation (24)) of the best particle at a maximum of 500 iterations) for variant 1, 5 and 6 of PSO using adaptive $\omega$

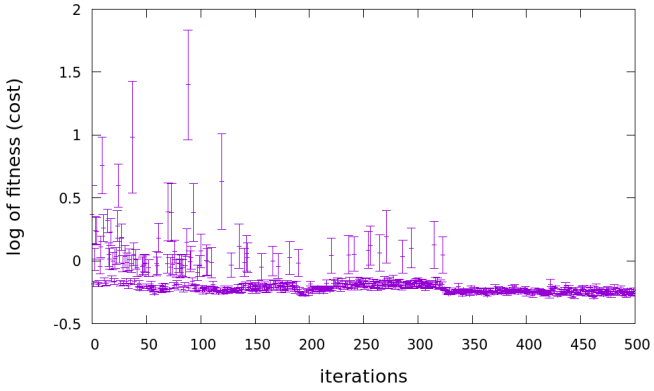
| Number of | Variant 1 | Variant 5 | Variant 6 |
|---|---|---|---|
| nodes | mean ($\pm SD$) | mean ($\pm SD$) | mean ($\pm SD$) |
| 9 | 0.4 ($\pm$0.006) | 0.4 ($\pm$0.0005) | 0.4 ($\pm$0.0004) |
| 16 | 0.4 ($\pm$0.01) | 0.4 ($\pm$0.04) | 0.4 ($\pm$0.0004) |
| 49 | 0.4 ($\pm$0.03) | 0.4 ($\pm$0.03) | 0.4 ($\pm$0.02) |

**TABLE 5.** Comparison of final results (value of the best particle at max iterations of 100 iterations) of variant 1, 5 and 6 of PSO using adaptive $\omega$

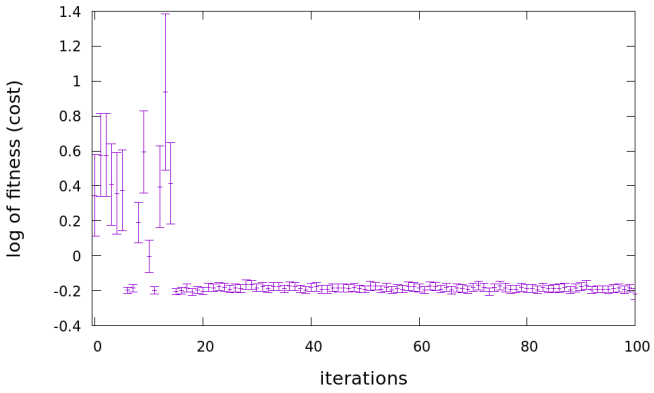
| Number of | Variant 1 | Variant 5 | Variant 6 |
|---|---|---|---|
| nodes | mean ($\pm SD$) | mean ($\pm SD$) | mean ($\pm SD$) |
| 9 | 0.4 ($\pm$0.08) | 0.4 ($\pm$0.03) | 0.4 ($\pm$0.02) |
| 16 | 0.4 ($\pm$0.02) | 0.4 ($\pm$0.05) | 0.4 ($\pm$0.01) |
| 49 | 0.4 ($\pm$0.03) | 0.4 ($\pm$0.03) | 0.4 ($\pm$0.02) |

### *E. COMPARISON BETWEEN ALL ALGORITHMS*

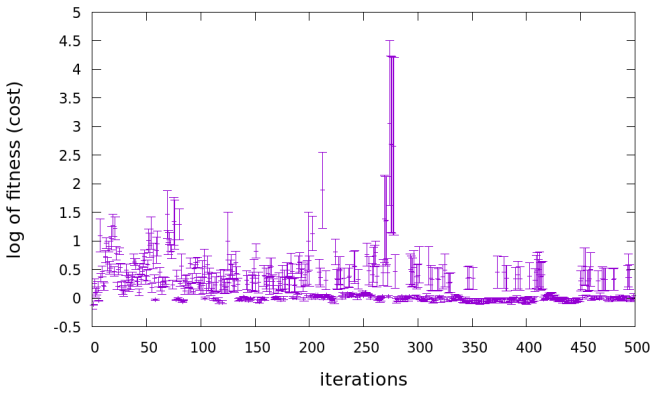
Tables 6 and 7 list the mean and standard deviations of the final results obtained from the 10 runs of each of the variations of the algorithms considered, after 100 and 50 iterations, respectively for GA, DE, and PSO; and 2000 and 1000 iterations, respectively for SA. The final result for the population-based algorithms is the cost of the best individual in the population by the specified iteration number for that run. For SA, we simply list the average of the last values obtained for 10 runs by 2000 and 1000 iterations,




(a) 9 nodes variant 1

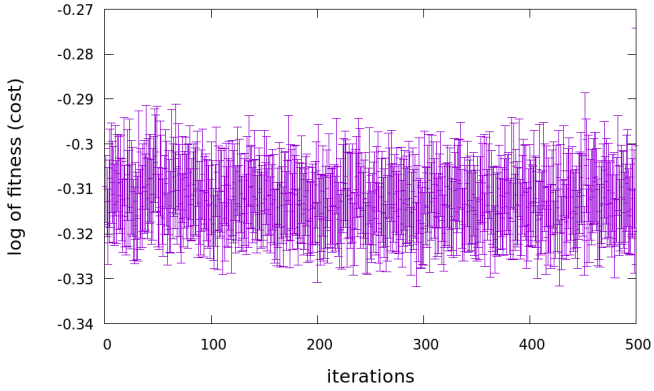



(b) 9 nodes variant 5

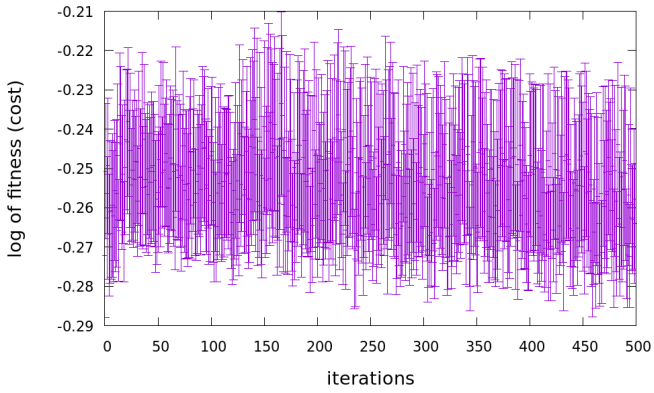



(c) 9 nodes variant 6

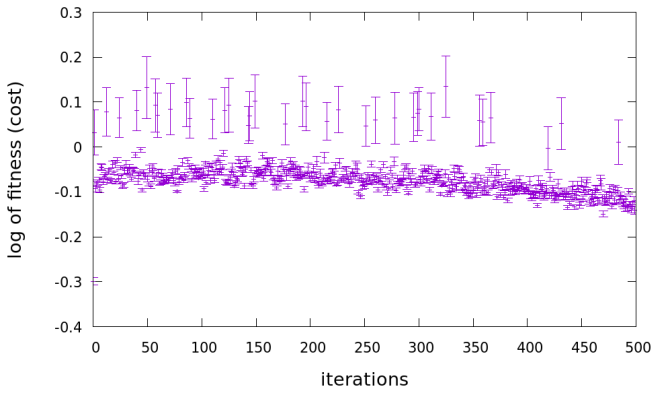
**FIGURE 12.** Mean with standard error of the natural logarithm (ln) of the mean cost of populations over 10 runs of PSO at each iteration over the running time (iterations) for a 9-node network. Each iteration represents 20 function evaluations for a population size of 20.




(a) 16 nodes variant 1




(b) 16 nodes variant 5

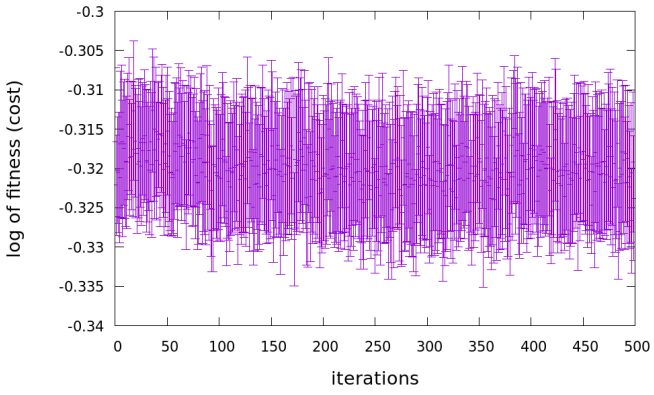



(c) 16 nodes variant 6

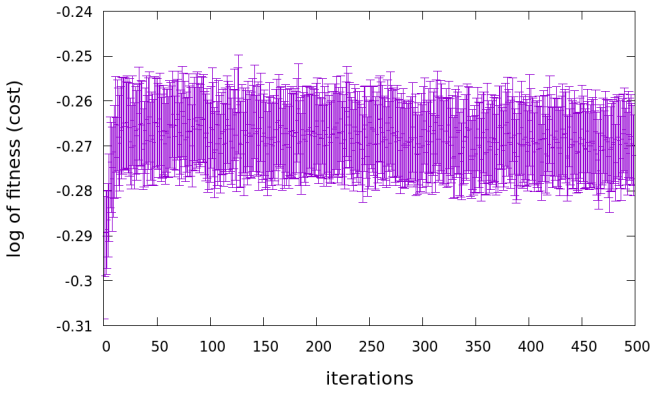
**FIGURE 13.** Mean with standard error of the natural logarithm (ln) of the mean cost of populations over 10 runs of PSO at each iteration over the running time (iterations) for a 16-node network. Each iteration represents 20 function evaluations for a population size of 20.

respectively. The 2000-iteration long run of SA is equivalent to a 100-iteration run of the population-based algorithms with 20 individuals, and 1000 iterations of SA is equivalent to 50 iterations of the population-based algorithms.

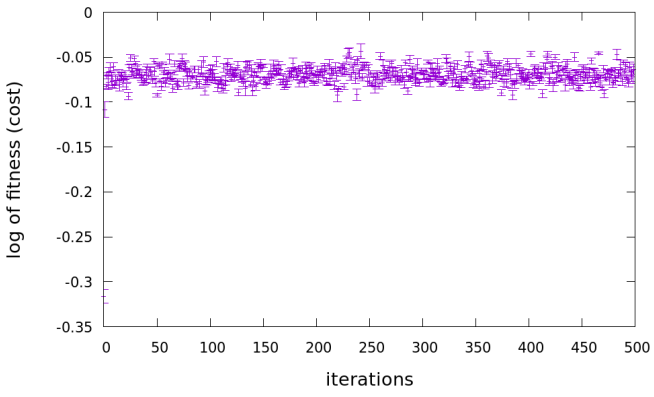
By considering Tables 6 and 7, we can see that DE is the clear winner, achieving very good end results with a very small standard deviation within 100 iterations as well as 50 iterations. There is no significant improvement between 50 and 100 iterations. As seen in Section VI-C, good results are achieved within as few as 20 iterations. GA also performed well. With a mutation rate of 0.5, it was able to find exactly the same results for all runs after 100 iterations for the 9-node network. We assume this is the actual optimal solution. After 50 iterations to 100 iterations, there was a negligible




(a) 49 nodes variant 1




(b) 49 nodes variant 5




(c) 49 nodes variant 6

**FIGURE 14.** Mean with standard error of the natural logarithm (ln) of the mean cost of populations over 10 runs of PSO at each iteration over the running time (iterations) for a 49-node network. Each iteration represents 20 function evaluations for a population size of 20.

difference in the solutions from this assumed optimal. While DE is the superior algorithm overall for this problem from these results, it was not able to achieve this final result with no deviation, although it did settle on the same solution within a very small deviation. However, the performance of GA deteriorates as the network size increases as compared with DE, and the performance after 50 iterations is inferior to that

**TABLE 6.** Comparison of final results obtained from SA (2000 iterations = 2000 function evaluations), and GA, DE, and PSO (100 iterations = 2000 function evaluations)

| | 9 nodes | 16 nodes | 49 nodes |
|---|---|---|---|
| Algorithm | mean ($\pm SD$) | mean ($\pm SD$) | mean ($\pm SD$) |
| SA | 0.5 ($\pm$0.1) | 0.4 ($\pm$0.05) | 5.2 ($\pm$7.8) |
| GA 0.5 | 0.4 ($\pm$0.0) | 0.4 ($\pm$0.0004) | 0.4 ($\pm$0.02) |
| GA 0.25 | 0.4 ($\pm$0.0005) | 0.4 ($\pm$0.0005) | 0.4 ($\pm$0.04) |
| DE | 0.4 ($\pm$0.0006) | 0.4 ($\pm$0.0006) | 0.4 ($\pm$0.0007) |
| PSO 1 | 0.4 ($\pm$0.08) | 0.4 ($\pm$0.02) | 0.4 ($\pm$0.03) |
| PSO 5 | 0.4 ($\pm$0.03) | 0.4 ($\pm$0.05) | 0.4 ($\pm$0.03) |
| PSO 6 | 0.4 ($\pm$0.02) | 0.4 ($\pm$0.02) | 0.4 ($\pm$0.02) |

**TABLE 7.** Comparison of final results obtained from SA (1000 iterations = 1000 function evaluations), and GA, DE, and PSO (50 iterations = 1000 function evaluations)

| | 9 nodes | 16 nodes | 49 nodes |
|---|---|---|---|
| Algorithm | mean ($\pm SD$) | mean ($\pm SD$) | mean ($\pm SD$) |
| SA | 2.0 ($\pm$1.8) | 2.2 ($\pm$1.8) | 16.2 ($\pm$12.7) |
| GA 0.5 | 0.4 ($\pm$0.0001) | 0.7 ($\pm$1.0) | 0.4 ($\pm$0.03) |
| GA 0.25 | 0.4 ($\pm$0.0005) | 0.4 ($\pm$0.0005) | 0.5 ($\pm$0.05) |
| DE | 0.4 ($\pm$0.0006) | 0.4 ($\pm$0.0006) | 0.4 ($\pm$0.0007) |
| PSO 1 | 0.4 ($\pm$0.07) | 0.4 ($\pm$0.02) | 0.4 ($\pm$0.03) |
| PSO 5 | 0.4 ($\pm$0.03) | 0.4 ($\pm$0.05) | 0.4 ($\pm$0.03) |
| PSO 6 | 0.4 ($\pm$0.005) | 0.4 ($\pm$0.01) | 0.4 ($\pm$0.02) |

of DE for the medium-size and larger networks, as we can see by looking at Table 7. While PSO also produces fairly good end results, the high likelihood of a good result within a small number of iterations seen in DE cannot also be expected from PSO because the variance is larger, and the convergence is less clear. Out of the PSO variants, variant 6 (FIPS) performs the best. While the mean final costs of the different PSO variants are similar, FIPS has the smallest variance, meaning that there is a slightly higher chance of obtaining this good average of 0.4. We also observe that PSO and DE are more robust to increasing network sizes than GA, as the results are still as good for the 9-node network as for the 49-node network. This is not the case for GA and SA. These algorithms show a marked deterioration in performance as the number of nodes in the network increases. In general, the performance of SA is inferior to the other algorithms. The final results are significantly higher than for the other algorithms, although we note that this is still a considerable improvement from the initial random solutions. The results produced by SA are still much better than the worst possible CA solution. SA is the least robust to increasing network size. We see that DE, in particular, can provide far superior solutions to SA, or any of the other population-based algorithms, within the same effective number of `mesh-sim` simulation runs (function evaluations). PSO and GA also perform fairly well.

We ran an additional set of experiments on a network with

**TABLE 8.** Comparison of final results obtained from SA (2000 iterations = 2000 function evaluations), and GA, PSO, and DE (100 iterations = 2000 function evaluations) for a topology of 49 nodes randomly placed on a disc

| Algorithm | mean | $\pm SD$ |
|---|---|---|
| SA | 0.7 | 0.1 |
| GA 0.5 | 0.6 | 0.04 |
| DE | 0.5 | 0.001 |
| PSO 1 | 0.5 | 0.02 |

**TABLE 9.** Friedman test statistics

| Node topology | Statistic (Q) | p-value |
|---|---|---|
| 9 nodes (grid) | 13.2 | 0.004 |
| 16 nodes (grid) | 15.6 | 0.001 |
| 49 nodes (grid) | 10.4 | 0.02 |
| 49 nodes (random in a disc) | 16.2 | 0.001 |

49 nodes randomly placed inside a disc, using a uniform distribution for the polar coordinates. This was to show the behaviour of the different algorithms for a more realistic topology. For these experiments, we ran each algorithm 10 times for 2000 function evaluations (2000 iterations of SA and 100 iterations for each of the population-based algorithms) and recorded the best final values of each run. Only one variation of each algorithm was used. A mutation rate of 0.5 was used for the GA, Variant 1 was used for PSO, and $F = 0.9, CR = 0.9$ for DE. The results are shown in Table 8. Here we see that SA performed significantly better for the random topology than it did for the grid topology. We still observe the best performance from DE, followed by PSO, then GA, and finally SA. Since we have rounded to one decimal place it is not visible, but DE has a final mean of 0.497 and PSO of 0.547, so DE does slightly outperform PSO. SA has the largest standard deviation by the end, GA converges slightly better than SA, PSO is better still, and finally DE has the lowest standard deviation among the final results of different runs. We also observe that the final values are not significantly different from those obtained using the grid topology. Thus we deduce that the grid topology is an adequate model for comparing the performance of CA optimisation algorithms.

Furthermore, we compared the algorithms using the Friedman test to determine whether the differences in the results obtained from the different algorithms are statistically significant. We ran each algorithm 10 times and recorded the best final cost value for each run. For the statistical test, we considered 2000 function evaluations in each case (100 iterations for the population-based algorithms). For the 9-node networks, the test statistic is 13.2 with a p-value of 0.004. For the 16-node networks, the $Q$ statistic is 15.61 with a p-value of 0.0014. For the 49-node grid network, the test statistic is 9.8 with p-value=0.02. Finally, for the 49-node random topology network, $Q = 16.2$ and $p = 0.001$. Since, for all of the considered network sizes, the p-value is less than 0.05, we can reject the null hypothesis that the results obtained from SA, GA, PSO and DE are the same, and conclude that there is a statistically significant difference in the results obtained using the different algorithms.

## VII. CONCLUSIONS AND RECOMMENDATIONS

This paper presents a new angle to the channel assignment problem in Wireless Mesh Networks (WMNs) – that of introducing Dynamic Spectrum Access (DSA). We provide metaheuristic solutions to the channel assignment problem in a WMN using DSA that find near-optimal channel allocations in the presence of external interference sources and avoid interference to Primary Users of the spectrum. We provide a novel algorithm used alongside the metaheuristic algorithms for optimisation. This algorithm of ours guarantees the feasibility of the CA solutions by ensuring that a) the interface constraint is met and b) connectivity is preserved in the network as much as possible. We evaluate the performance using a new simulation framework that we developed in ns3. This simulation framework models a WMN with DSA, which we publicly shared for others to use. The performance of four algorithms, namely Simulated Annealing (SA), Genetic Algorithm (GA), Differential Evolution (DE), and Particle Swarm Optimisation (PSO), are compared for finding optimising channel assignment.

We observe very good performance by the DE and GA algorithms. We note in particular that DE scales to larger networks effectively, without needing to increase the run time, and that DE has low computational complexity. At the same time, GA is less robust to expanding the network size and has high computational complexity by comparison with the other algorithms. While PSO does not display equally clear convergence within the number of iterations considered, the final results are still good. From repeating the experiments we find that the standard deviation of the final results is about an order of magnitude larger for PSO than for DE, but it is still insignificant, e.g., 0.07 vs 0.0006. Additionally, PSO has good computational complexity. SA does not scale well, but is still able to provide improved solutions and show convergence within the considered number of simulation instances. SA has the advantage of low computational complexity.

Considering our results, in practice, we would wholeheartedly recommend the DE algorithm as being best suited to this problem. It can achieve very good results within as few as 20 iterations. None of the other three tested algorithms is able to give this guarantee within so few iterations.

## ACKNOWLEDGMENT

The authors thank Dr. Anna Bosman of the University of Pretoria and Dr. Clement Nyirenda of the University of the Western Cape for their valuable input on the intricacies of PSO. Gratitude also goes to the Telkom Foundation for the provision of funding that contributed to this work.

## REFERENCES


[1] "CBRS, SAS and Spectrum Sharing: The Complete Guide." [Online]. Available: https://blinqnetworks.com/cbrs-sas-spectrum-sharing-guide/

[2] "Wi-Fi Alliance® furthers Automated Frequency Coordination specification and compliance development to accelerate Wi-Fi 6E," 2021. [Online]. Available: https://www.wi-fi.org/news-events/newsroom/wi-fi-alliance-furthers-automated-frequency-coordination-specification-and

[3] R. Forum, "Dynamic Spectrum Sharing for 5G NR and 4G LTE Coexistence," 11 2020. [Online]. Available: https://www.rcrwireless.com/20201117/opinion/dynamic-spectrum-sharing-for-5g-nr-and-4g-lte-coexistence

[4] "iNethi," 2021. [Online]. Available: www.inethi.org.za/

[5] "Zenzeleni Community Networks," 2021. [Online]. Available: zenzeleni.net/

[6] "Altermundi," 2021. [Online]. Available: altermundi.net/

[7] M. Garey and D. Johnson, *Computers and intractability: A Guide to the Theory of NP-Completeness*, V. Klee, Ed. New York: W.H. Freeman and company, 1979.

[8] W. K. Hale, "Frequency assignment: Theory and applications," in *Proceedings of the IEEE*, vol. 68, no. 12, 1980, pp. 1497–1514.

[9] M. Fitch, M. Nekovee, S. Kawade, K. Briggs, and R. MacKenzie, "Wireless service provision in TV white space with cognitive radio technology: A telecom operator's perspective and experience," *IEEE Communications Magazine*, vol. 49, no. 3, pp. 64–73, 2011.

[10] S. Schley, "TV White Space gets a piece of the auction," 2016. [Online]. Available: https://www.unh.edu/broadband/tv-white-space-gets-piece-auction

[11] ICASA, "DRAFT REGULATIONS ON THE USE OF TELEVISION WHITE SPACES," Sandton, pp. 703–718, 2017.

[12] M. R. Garey and D. S. Johnson, "The Complexity of Near-Optimal Graph Coloring," *Journal of the ACM (JACM)*, vol. 23, no. 1, pp. 43–49, 1976.

[13] R. M. Karp, "Reducibility among Combinatorial Problems," in *Complexity of Computer Computations.*, B. J. Miller R.E., Thatcher J.W., Ed. Boston, MA: Springer, 1972, pp. 85–103.

[14] I. F. Akyildiz and X. Wang, "A survey on wireless mesh networks," *IEEE Communications Magazine*, vol. 43, no. 9, pp. 23–30, 2005.

[15] S. Sampaio, P. Souto, and F. Vasques, "A review of scalability and topological stability issues in IEEE 802.11s wireless mesh networks deployments," *International Journal of Communication Systems*, vol. 29, pp. 671–693, 2016.

[16] W. K. Wong and C. I. Ming, "A Review on Metaheuristic Algorithms: Recent Trends, Benchmarking and Applications," in *2019 7th International Conference on Smart Computing and Communications, ICSCC 2019*. IEEE, 2019, pp. 1–5.

[17] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by Simulated Annealing," *Science, New Series*, vol. 220, no. 4598, pp. 671–680, 1983.

[18] Y. Nourani and B. Andresen, "A comparison of simulated annealing cooling strategies," *Journal of Physics A: Mathematical and General*, vol. 31, no. 41, pp. 8373–8385, 1998.

[19] J. Geweke and H. Tanizaki, "Bayesian estimation of state-space models using the Metropolis Hastings algorithm within Gibbs sampling," *Computational statistics and data analysis*, vol. 37, no. 2, pp. 151–170, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167947301000093

[20] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[21] Lichtblau, "Differential Evolution in Discrete Optimization," *International Journal of Swarm Intelligence and Evolutionary Computation*, vol. 1, pp. 1–10, 2012.

[22] J. Kennedy and R. C. Eberhart, "Particle Swarm Optimization," *Studies in Computational Intelligence*, vol. 927, pp. 1942–1948, 1995.

[23] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE Conference on Evolutionary Computation, ICEC*, no. February 2015, Singapore, 1998, pp. 69–73.

[24] Y. L. Zheng, L. H. Ma, L. Y. Zhang, and J. X. Qian, "On the convergence analysis and parameter selection in particle swarm optimization," *International Conference on Machine Learning and Cybernetics*, vol. 3, no. November, pp. 1802–1807, 2003.

[25] J. Kennedy, "Bare bones particle swarms," *2003 IEEE Swarm Intelligence Symposium, SIS 2003 - Proceedings*, pp. 80–87, 2003.

[26] J. Kennedy and R. C. Eberhart, "Discrete binary version of the particle swarm algorithm," *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 5, pp. 4104–4108, 1997.

[27] D. Anghinolfi and M. Paolucci, "A new discrete particle swarm optimization approach for the single-machine total weighted tardiness scheduling problem with sequence-dependent setup times," *European Journal of Operational Research*, vol. 193, pp. 73–85, 2009.

[28] X. Li, H. Xu, and Z. Cheng, "One improved discrete particle swarm optimization based on quantum evolution concept," *Proceedings - International Conference on Intelligent Computation Technology and Automation, ICICTA 2008*, vol. 1, pp. 96–100, 2008.

[29] X. Wang and L. Tang, "A discrete particle swarm optimization algorithm with self-adaptive diversity control for the permutation flowshop problem with blocking," *Applied Soft Computing Journal*, vol. 12, no. 2, pp. 652–662, 2012. [Online]. Available: http://dx.doi.org/10.1016/j.asoc.2011.09.021

[30] K. Chowdhury and I. Akyildiz, "Cognitive Wireless Mesh Networks with Dynamic Spectrum Access," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 1, pp. 168–181, 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4413149

[31] C. S. Xin, S. Ullah, M. Song, Z. Wu, Q. Gu, and H. Cui, "Throughput oriented lightweight near-optimal rendezvous algorithm for cognitive radio networks," *Computer Networks*, vol. 137, pp. 49–60, 2018.

[32] Y. Qin, J. Zheng, X. Wang, H. Luo, H. Yu, X. Tian, and X. Gan, "Opportunistic scheduling and channel allocation in MC-MR cognitive radio networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 7, pp. 3351–3368, 2014.

[33] A. P. Subramanian, H. Gupta, and S. R. Das, "Minimum interference channel assignment in multi-radio wireless mesh networks," *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON*, pp. 481–490, 2007.

[34] A. U. Chaudhry, R. H. Hafez, and J. W. Chinneck, "On the impact of interference models on channel assignment in multi-radio multi-channel wireless mesh networks," *Ad Hoc Networks*, vol. 27, pp. 68–80, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.adhoc.2014.11.019

[35] A. U. Chaudhry, J. W. Chinneck, and R. H. Hafez, "Fast heuristics for the frequency channel assignment problem in multi-hop wireless networks," *European Journal of Operational Research*, vol. 251, no. 3, pp. 771–782, 2016.

[36] A. U. Chaudhry, R. H. Hafez, and J. W. Chinneck, "Realistic interference-free channel assignment for dynamic wireless mesh networks using beamforming," *Ad Hoc Networks*, vol. 51, pp. 21–35, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.adhoc.2016.08.001

[37] Y. Y. Chen and C. Chen, "Simulated annealing for interface-constrained channel assignment in wireless mesh networks," *Ad Hoc Networks*, vol. 29, pp. 32–44, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.adhoc.2015.01.019

[38] S. Sridhar, J. Guo, and S. Jha, "Channel Assignment in Multi-Radio Wireless Mesh Networks : A Graph-Theoretic Approach," in *2009 First International Communication Systems and Networks and Workshops*, Bangalore, 2009, pp. 1–10.

[39] A. Pal and A. Nasipuri, "JRCA: A joint routing and channel assignment scheme for wireless mesh networks," in *Conference Proceedings of the IEEE International Performance, Computing, and Communications Conference*. IEEE, 2011, pp. 1–8.

[40] Y. Ding, Y. Huang, G. Zeng, and L. Xiao, "Channel assignment with partially overlapping channels in wireless mesh networks," in *Proceedings of the 4th Annual International Conference on Wireless Internet*, Maui, 2008, pp. 1–9.

[41] N. Balusu, S. Pabboju, and G. Narsimha, "An Intelligent Channel Assignment Approach for Minimum Interference in Wireless Mesh Networks Using Learning Automata and Genetic Algorithms," *Wireless Personal Communications*, vol. 106, no. 3, pp. 1293–1307, 2019. [Online]. Available: https://doi.org/10.1007/s11277-019-06214-3

[42] H. Cheng and S. Yang, "Joint QoS multicast routing and channel assignment in multiradio multichannel wireless mesh networks using intelligent computational methods," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1953–1964, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.asoc.2010.06.011

[43] A. P. Subramanian, R. Krishnan, S. R. Das, and H. Gupta, "Minimum Interference Channel Assignment in Multi-Radio Wireless Mesh Networks," in *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, vol. 1, 2007, pp. 481–490.

[44] X. Zhuang, H. Cheng, N. Xiong, and L. T. Yang, "Channel assignment in multi-radio wireless networks based on PSO algorithm," in *2010 5th*

*International Conference on Future Information Technology, FutureTech 2010 - Proceedings*. Busan: IEEE, 2010.

[45] S. Ghosh, A. Konar, and A. Nagar, "Dynamic channel assignment problem in mobile networks using particle swarm optimization," *Proceedings - EMS 2008, European Modelling Symposium, 2nd UKSim European Symposium on Computer Modelling and Simulation*, pp. 64–69, 2008.

[46] H. M. Abdelsalam, H. S. Hamza, A. M. Al-Shaar, and A. S. Hamza, "On the use of particle swarm optimization techniques for channel assignments in cognitive radio networks," in *Multidisciplinary Computational Intelligence Techniques: Applications in Business, Engineering, and Medicine*. Hershey: Information Science Reference (an imprint of IGI Global), 2012, no. July, pp. 202–214.

[47] M. Chakraborty, R. Chowdhury, J. Basu, R. Janarthanan, and A. Konar, "A particle swarm optimization-based approach towards the solution of the dynamic channel assignment problem in mobile cellular networks," in *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, 2008.

[48] S. Sakamoto, K. Ozera, A. Barolli, M. Ikeda, L. Barolli, and M. Takizawa, "Implementation of an intelligent hybrid simulation systems for WMNs based on particle swarm optimization and simulated annealing: performance evaluation for different replacement methods," *Soft Computing*, vol. 23, no. 9, pp. 3029–3035, 2019. [Online]. Available: https://doi.org/10.1007/s00500-017-2948-1

[49] M. R. Rai, S. Vahid, and K. Moessner, "SINR based topology control for multihop wireless networks with fault tolerance," in *2015 IEEE 81st Vehicular Technology Conference*, vol. 2015, 2015, pp. 1–6.

[50] M. Da Silva Maximiano, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, and J. M. Sánchez-Pérez, "Solving the frequency assignment problem with differential evolution," *2007 15th International Conference on Software, Telecommunications and Computer Networks, SoftCOM 2007*, pp. 119–123, 2007.

[51] J. A. G.-P. J. M. S.-P. Marisa da Silva Maximiano, Miguel A. Vega-Rodriquez, "A Hybrid Differential Evolution Algorithm to Solve a Real-World Frequency Assignment Problem," in *Proceedings of the International Multiconference on Computer Science and Information Technology*, 2008, pp. 201–205.

[52] e. a. Latif, "Channel assignment using differential evolution algorithm in cognitive radio networks," *International Journal of Advanced and Applied Sciences*, vol. 4, no. 8, pp. 160–166, 2017.

[53] K. K. Anumandla, B. Akella, S. L. Sabat, and S. K. Udgata, "Spectrum allocation in cognitive radio networks using multi-objective differential evolution algorithm," in *2nd International Conference on Signal Processing and Integrated Networks, SPIN 2015*. IEEE, 2015, pp. 264–269.

[54] R. Maliwatu, "A new connectivity strategy for Wireless Mesh Networks using Dynamic Spectrum Access," Ph.D. dissertation, University of Cape Town, 2020.

[55] Aruba, "Airmatch." [Online]. Available: https://www.arubanetworks.com/techdocs/ArubaOS_81_Web_Help/Content/ArubaFrameStyles/ARM/mCell.htm

[56] S. Cho, "SINR-Based MCS Level Adaptation in CSMA/CA Wireless Networks to Embrace IoT Devices," *Symmetry*, vol. 9, no. 10, p. 236, 2017.

[57] P. Fuxjaeger and S. Ruehrup, "Validation of the NS-3 Interference Model for IEEE802.11 Networks," *Proceedings - 2015 8th IFIP Wireless and Mobile Networking Conference, WMNC 2015*, no. October 2015, pp. 216–222, 2016.

[58] "natzlob/ns-3-dev-git," 2021. [Online]. Available: https://github.com/natzlob/ns-3-dev-git

[59] S. Zhang, A. S. Hafid, H. Zhao, and S. Wang, "Cross-Layer Rethink on Sensing-Throughput Tradeoff for Multi-Channel Cognitive Radio Networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 10, pp. 6883–6897, 2016.

[60] IEEE, "Draft Standard for Wireless Regional Area Networks Part 22 : Cognitive Wireless RAN Medium Access Control ( MAC ) and Physical Layer ( PHY ) specifications : Policies and procedures for operation in the TV Bands," 2010.

[61] "cplusplus.com," p. std::random_shuffle, 2021. [Online]. Available: http://www.cplusplus.com/reference/algorithm/random_shuffle/

[62] A. V. Kononova, D. W. Corne, P. De Wilde, V. Shneer, and F. Caraffini, "Structural bias in population-based algorithms," *Information Sciences*, vol. 298, pp. 468–490, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2014.11.035

[63] "cplusplus.com," p. std::sort, 2021. [Online]. Available: http://www.cplusplus.com/reference/algorithm/sort/

[64] "cplusplus.com," p. std::find, 2021. [Online]. Available: https://www.cplusplus.com/reference/algorithm/find/

[65] "cplusplus.com," p. std::map::insert, 2021. [Online]. Available: https://www.cplusplus.com/reference/map/map/insert/

[66] S. Mirjalili, "Genetic algorithm," in *Evolutionary Algorithms and Neural Networks. Studies in Computational Intelligence*. Springer, Cham, 2019, vol. 780, pp. 43–55.

[67] K. Deb and S. Agrawal, "Understanding Interactions Among Genetic Algorithm Parameters," *Foundations of Genetic Algorithms*, vol. 5, pp. 265–286, 1999.

[68] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading MA: Addison-Wesley, 1989.

[69] R. L. Haupt, "Optimum population size and mutation rate for a simple real genetic algorithm that optimizes array factors," in *IEEE Antennas and Propagation Society International Symposium. Transmitting Waves of Progress to the Next Millennium. 2000 Digest.* IEEE, 2000, pp. 1034–1037.

[70] T. Bäck, "Optimal mutation rates in genetic search," *Proceedings of the 5th International Conference on Genetic Algorithms*, vol. 28, pp. 2–8, 1993.

[71] A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, and V. B. Prasath, "Choosing mutation and crossover ratios for genetic algorithms-a review with a new dynamic approach," *Information (Switzerland)*, vol. 10, no. 12, 2019.

[72] M. Georgioudakis and V. Plevris, "A Comparative Study of Differential Evolution Variants in Constrained Structural Optimization," *Frontiers in Built Environment | www.frontiersin.org*, vol. 1, p. 102, 2020. [Online]. Available: www.frontiersin.org

[73] I. C. Trelea, "The particle swarm optimization algorithm: Convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, no. 6, pp. 317–325, 2003.

[74] M. Jiang, Y. P. Luo, and S. Y. Yang, "Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm," *Information Processing Letters*, vol. 102, no. 1, pp. 8–16, 2007.

[75] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," *Proceedings of the 2007 IEEE Swarm Intelligence Symposium, SIS 2007*, no. Sis, pp. 120–127, 2007.

[76] W. Jiao, G. Liu, and D. Liu, "Elite Particle Swarm Optimization with Mutation," *2008 Asia Simulation Conference - 7th International Conference on System Simulation and Scientific Computing, ICSC 2008*, no. 3, pp. 800–803, 2008.

[77] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[78] G. Xu and G. Yu, "On convergence analysis of particle swarm optimization algorithm," *Journal of Computational and Applied Mathematics*, vol. 333, pp. 65–73, 2018. [Online]. Available: https://doi.org/10.1016/j.cam.2017.10.026

NATASHA ZLOBINSKY was born in Durban, South Africa in 1989. She received her BSc in Electrical Engineering in 2011 and her MSc in Telecommunications Engineering with Distinction in 2017, both from the University of the Witwatersrand, Johannesburg, South Africa. She worked as an Engineer for the power utility Eskom 2012-2015 and in the Future Wireless Network technologies research group at the CSIR Meraka Institute from 2015 to 2018, both in South Africa. She is currently a device software engineer at Aruba User Experience Insights (formerly Cape Networks) in Cape Town, South Africa, and is pursuing a PhD in Computer Science at the University of Cape Town, where she is a member of the Net4D research group. Her areas of research interest are centred around wireless communications, and include the Internet of Things, alternative spectrum and Dynamic Spectrum Access, Wireless Mesh Networks, Information and Communications Technology for Development (ICT4D), Software Defined Radio, and Machine Learning and optimisation algorithms.

DAVID L. JOHNSON serves as an adjunct Senior Lecturer in the Computer Science Department at the University of Cape Town in the ICT4D lab, a Senior Research Associate at Research ICT Africa and a Telecommunications Consultant at the Vernonburg group. He was recently an IT Policy fellow at the Centre for Information and Technology at Princeton University. He has published 70 articles in the general area of wireless connectivity and ICT for development and a book on TV White Space technology. David earned a B.Eng in Electronic Engineering from the University of Cape Town, South Africa. He completed his M.Eng. in Computer Engineering at the University of Pretoria, South Africa and a M.Sc and Ph.D. in Computer Science from the University of California, Santa Barbara, U.S.A. on Internet architectures for rural developing regions.

AMIT KUMAR MISHRA (Senior Member, IEEE) received a Ph.D. degree in radar signal processing from The University of Edinburgh, Edinburgh, U.K., in 2006. Since his Ph.D., he has been an active researcher in the domain of sensor design, radar, applied machine learning, and frugal innovation. He is currently a Professor with the Department of Electrical Engineering, University of Cape Town, Cape Town, South Africa. His Google Scholar based H-index is 14. He has more than 150 peer-reviewed publications and holds five patents.

ALBERT A. LYSKO is an award-winning engineer, researcher and innovator. He is a Principal Researcher with the Council for Scientific and Industrial Research (CSIR), South Africa. Dr Lysko has worked in both academia and industry, in both Europe and Africa. His research focus has covered numerical electromagnetics, smart antennas, dynamic spectrum access and is now shifting into 5G and 6G. While at CSIR, Dr Lysko's leading experimental research in television white spaces (TVWS) provided Internet to over 20,000 users in three countries and enabled setting up the South African national TVWS regulation and contributed to TVWS regulations in other African countries and USA. He has authored three patents, a book, two book chapters, and over 100 research papers, popular science, and news articles. He holds 3 Best Paper and several professional awards. As a volunteer for the Institution of Electrical and Electronics Engineers (IEEE), Dr Lysko has organised over 100 events and three international conferences. Dr Lysko has numerous IEEE awards for volunteering. Under his leadership, IEEE South Africa received its first global IEEE MGA award. Dr Lysko is a Fellow of South African Institute of Electrical Engineers.