Parallel 3-Parent Genetic Algorithm with Application to Routing in Wireless Mesh Networks

Amar Singh, Shakti Kumar, Ajay Singh, and Sukhbir S. Walia

1 Introduction

Genetic algorithms (GAs) are widely used computer-based search and optimization algorithms based on the mechanics of natural genetics and natural selection [1–5]. In the decade between 1950 and 1960 many researchers worked on evolutionary systems with the idea that evolution could be used as optimization approach for many engineering problems [4]. J. Holland introduced the concept of genetic algorithm in 1960 [5]. Usually genetic algorithms are based on two-parent genetic processes; however, some literature on multi-parent recombination can also be found in [6–9]. Mühlenbein and Voigt [6] presented the concept of gene pool recombination (GPR), and applied it to find solutions in a discrete domain. Eiben and Van Kemenade [7] proposed the concept of diagonal crossover as generalization of uniform crossover in GA and applied it to numerical optimization problems. Wu et al. [8] proposed multi-parent orthogonal recombination and applied it to find out the identity of an unknown image contour. The crossover operators used in those areas enabled significant improvements in search ability, although improvements were found to be highly problem dependent. Eiben et al. [9] proposed two multi-

A. Singh
Lovely Professional University, Phagwara, Punjab, India

S. Kumar (✉)
Panipat Institute of Engineering & Technology, Panipat, Haryana, India

A. Singh
Hochschule Wismar, University of Applied Sciences, Technology, Business and Design, Wismar, Germany

S. S. Walia
IK Gujral Punjab Technical University, Jalandhar, Punjab, India

S. Subair, C. Thron (eds.), *Implementations and Applications of Machine Learning*,
Studies in Computational Intelligence 782,
https://doi.org/10.1007/978-3-030-37830-1_1

parent recombination mechanisms, namely gene scanning and diagonal crossover. They extensively tested their multi-parent algorithm on a variety of problems in numerical optimization, constrained optimization (traveling salesman problem) and constraint satisfaction (graph coloring). Compared to two-parent recombination, their algorithm achieved superior performance in optimizing the first four test functions of De Jong. For other problems the results were mixed; multi-parent crossover sometimes performed better and at times worse than classical two-parent recombination.

The parallel three-parent genetic algorithm presented in this chapter is based upon three-parent genetic processes in medical science and is very different from the approaches available in the literature. In medical science, a three-parent process has been used to prevent mitochondrial diseases in children of mothers with defective mitochondria [10–13]. In 2015, Dr. John Zhang and his team at the New Hope Fertility Center in New York City replaced the nucleus of a donor's egg cell with the nucleus of original mother, which was then fertilized with the father's sperm and implanted in the mother. The child that was subsequently born inherited mitochondrial DNA from the donor, besides nuclear DNA from the father and mother [13]. The concept of this three-parent genetic process is shown in Fig. 1. The P3PGA algorithm is in some respects a mathematical analogy of this practical process.

This chapter includes two different but related research investigations. The first is an evaluation of the performance of P3PGA on functions from an established test suite and comparison with other well-known recent algorithms. The second investigation concerns the application of P3PGA to an important and challenging practical problem, namely routing in wireless mesh networks (WMNs) [14].

This chapter is organized into six sections. Section 1 presents the motivation for this work; Section 2 discusses the working of P3PGA algorithm; Section 3 describes

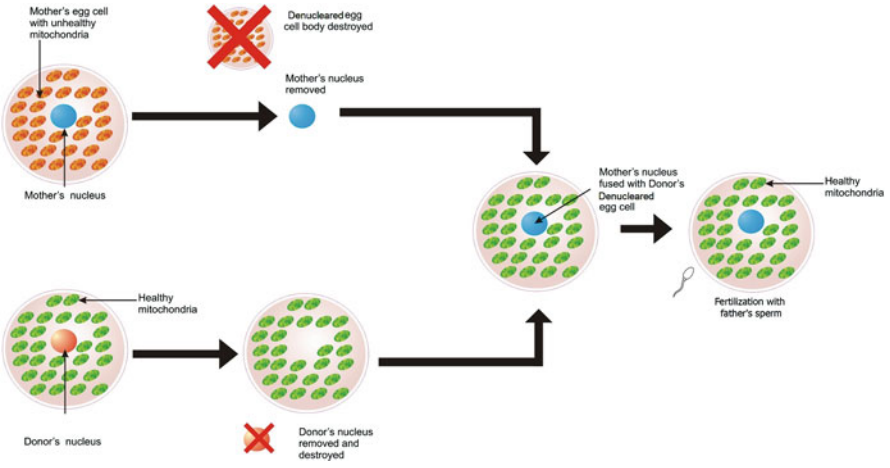

Fig. 1 3-Parent Process adapted from Zhang et al. [13]

the simulation and performance of P3PGA on the 2014 Congress on Evolutionary Computation (CEC-2014) test bench suite compared to 16 other algorithms; Section 4 proposes a P3PGA-based minimal cost route evaluation approach for WMNs; Section 5 presents its implementation and performance on WMNs as well as a performance comparison with eight other algorithms found in the literature; and Sect. 6 summarizes conclusions.

2 P3PGA Algorithm

The Parallel 3-Parent Genetic Algorithm (P3PGA) is a multi-population algorithm in which evolution process takes place on several populations in parallel. It is based upon the single-population three-parent genetic algorithm (3PGA) [15, 16].

A pseudocode for the proposed P3PGA approach is as given in the listing below in Algorithm 1. In this algorithm we initially create several populations of equal size: these are called 2-parent populations. To generate a 3-parent population from a 2-parent population we applied the "mitochondrial change" to each individual by adding a small random number to each gene of each individual in the population. The fittest individual (best solution) within a population is called the *local elite* of that specific population. The best of all local elites is called the *global elite*. All genes for all individuals in all the populations are moved towards the corresponding gene of the global best candidate solution with low probability. This guides every candidate solution towards the global best solution.

For the better understanding of the algorithm, we describe its operation on a very simple example where we wish to search a 4-digit quantity such that the sum of 4 digits is maximal, given that each digit is an integer between 0 and 9 (the obvious answer is 9999). We proceed in such a way that the reader will be able to track progress towards this solution as the algorithm progresses. The algorithm parameters are chosen as shown in Table 1.

Step 1 Randomly create 3 two-parent populations, where each population consists of three candidate solutions and each candidate solution is an array of four digits (genes) having the form (d_1, d_2, d_3, d_4):

TwoParentPop(1): [(3,6,1,7), (5,2,6,3), (4,6,5,3)]
TwoParentPop(2): [(5,3,6,2), (5,6,7,8), (8,2,7,6)]
TwoParentPop(3): [(7,4,5,1), (5,6,1,3), (3,8,4,9)]

We may rewrite these three populations in matrix format, for example:

$$\text{TwoParentPop}(1) = \begin{bmatrix} 3 & 6 & 1 & 7 \\ 5 & 2 & 6 & 3 \\ 4 & 6 & 5 & 3 \end{bmatrix}$$

TwoParentPop(2) and TwoParentPop(3) may be expressed similarly as

Algorithm 1: P3PGA algorithm

begin

Generate NP populations each of size N candidates randomly, every candidate consisting of NG genes;

for gen = 1:Number of generations **do**

  for i = 1:NP **do**

    Effect Mitochondrial Change to i^{th} 2-parent (2-P) population to Generate an i^{th} 3-Parent (3-P) population

    Combine the 2-P and 3-P populations and select the N best individuals.

    Find and record the globally best solution.

    Generate a new 2-P population using general genetic process (using GA)

      (a) Select fit individuals for recombining/breeding.

      (b) With high probability recombine parents/perform cross-over.

      (c) With low probability mutate offspring.

      (d) Select the N best individuals from among parents and offspring...

    Check bounds violation and correct if needed.

end for (i)

Check the fitness of all the individuals of all the populations and select/update the globally best g_{best} candidate and its fitness;

for i = 1: NP **do**

  for j = 1: N **do**

    for k = 1: NG **do**

      With a fixed small probability replace gene k of individual j in population i with (individual(j,k) + g_{best}(j,k))/2;

    end for (k)

  end for (j)

end for (i)

end for (gen)

end

Table 1 Parameter values for illustrative example of P3PGA

Parameter description	Symbol	Value
Number of populations	NP	3
Population size	N	3
Number of genes in each individual	NG	4
Gene values	d_j ($j = 1 \ldots 4$)	0 thru 9
Crossover probability		0.9
Mutation probability		0.1

$$\text{TwoParentPop}(2) = \begin{bmatrix} 5 & 3 & 6 & 2 \\ 5 & 6 & 7 & 8 \\ 8 & 2 & 7 & 6 \end{bmatrix}$$

and

$$\text{TwoParentPop}(3) = \begin{bmatrix} 7 & 4 & 5 & 1 \\ 5 & 6 & 1 & 3 \\ 3 & 8 & 4 & 9 \end{bmatrix}$$

Step 2 For each gene of every individual in each of the populations, effect a mitochondrial change by adding a random number to each gene. In our case, each gene change is generated as a uniformly distributed random integer between -2 and 2. This may be implemented by generating a change matrix for each population, adding the change matrix to the population matrix, and truncating so that the gene values remain within the range from 0 to 9. For example, suppose the change matrix for the first population is given by:

$$\text{Change}(1) = \begin{bmatrix} 2 & 0 & -2 & 1 \\ 1 & -1 & 2 & 1 \\ 0 & 2 & 2 & 2 \end{bmatrix}$$

Let us further assume that the randomly generated mitochondrial changes for the second and third populations are as given below:

$$\text{Change}(2) = \begin{bmatrix} -2 & 0 & -1 & 1 \\ 1 & 2 & 2 & -1 \\ 0 & 2 & 2 & 2 \end{bmatrix}; \quad \text{Change}(3) = \begin{bmatrix} -2 & 0 & -1 & 2 \\ 1 & 2 & 2 & -1 \\ 0 & 1 & -2 & 2 \end{bmatrix}$$

Using 2-parent population and Change matrices for each population we compute the 3-parent populations as follows:

$$\text{3\_Parent\_Population}(n) = \text{2\_Parent\_Population}(n) + \text{Change}(n)$$

Whenever this formula produces an entry in 3_Parent_Population(n) that is less than 0 or greater than 9, it is replaced with 0 or 9, respectively.

Step 3 The three-parent populations are then combined with their corresponding two-parent populations to form populations that are twice as large. In our example, these combined populations are given by three 6×3 matrices (matrix rows are separated by semicolons)

Population 1: [3 6 1 7; 5 2 6 3; 4 6 5 3; 5 6 0 8; 6 1 8 4; 4 8 7 5]
Population 2: [5 3 6 2; 5 6 7 8; 8 2 7 6; 3 3 5 3; 6 8 9 7; 8 4 9 8]
Population 3: [7 4 5 1; 5 6 1 3; 3 8 4 9; 5 4 4 3; 6 8 3 2; 3 9 2 9]

We choose the three individuals for each population with highest fitness, which are then sorted in decreasing order of fitness. Table 2 shows the results of this operation.

Table 2 Populations, individuals, and their Fitness

Pop. No.	Individual No.	Individual	Fitness	Pop. No.	Individual No.	Individual	Fitness
1	1	4 8 7 5	24	3	1	3 8 4 9	24
	2	5 6 0 8	19		2	3 9 2 9	23
	3	6 1 8 4	19		3	6 8 3 2	19
2	1	6 8 9 7	30				
	2	8 4 9 8	29				
	3	5 6 7 8	26				

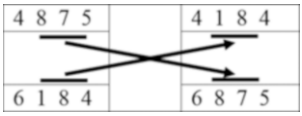

Fig. 2 Single-point crossover operation

Step 4 The conventional genetic algorithm operations of recombination (crossover) and mutation are now performed within the resulting populations.

Step 4a. Usually fitness of an individual is used as selection criterion for crossover. Various selection strategies may be used, including roulette wheel selection, stochastic universal sampling, truncation selection, tournament selection, and so on (interested reader may refer to [17–19] for details). We have used stochastic universal sampling (SUS) for selection of parents for recombination. In our example, we may suppose that individuals 1 and 3 in population 1 are selected, and a crossover operation is performed on the two individuals with high probability (usually between 0.75 and 0.9). A typical example of a crossover operation is shown in Fig. 2.

The operation shown is a single-point crossover with crossover point after first gene. Crossover can be single point, two point, or n-point. If crossover does not take place, then both individuals are passed on as offspring.

Step 4b. Following crossover, mutation is performed. In the case of real-valued genes, one way to perform mutation involves replacing the current gene with a randomly generated value from within the universe of discourse of that gene with a low probability. Both the probability of mutating a gene (mutation rate) and the distribution of changes for each mutated gene must be specified. For example, the change may be determined as a uniform random number in an interval $[-a, a]$. In our example, we have taken $a = 2$ and the mutated genes are rounded off to the nearest integer.

Step 4c. Following crossover and mutation, the offspring are grouped together with their parents for each population, and the three fittest from each population are chosen as the next generation. Note each generation of each population always has the same number of individuals (equal to N, which is one of the algorithm's basic parameters). Let us assume that the individuals 4 8 7 5 and 5 6 0 8 were selected for recombination assuming the crossover point was after first two genes. Thus, the combination produced two offspring, i.e., 4 8 0 8 and 5 6 7 5. Let us further assume that individual 6 1 8 4 was passed as it is as an offspring.

Table 3 Evolving new population after crossover and mutation

Current population	Offspring (after crossover and mutation)	New population (weak individuals replaced with stronger offspring)
4 8 7 5 (24)	4 8 0 7 (19)	5 8 7 5 (25)
5 6 0 8 (19)	5 8 7 5 (25)	4 8 7 5 (24)
6 1 8 4 (19)	6 0 8 4 (18)	5 6 0 8 (19)

Table 4 Local best (elites) and global best

Population number	Local best (ℓ_{best})/Elite	Fitness
1	5 8 7 5	25
2	6 8 9 7	30
3	5 9 2 9	25

Let us further assume that the mutation operator mutated first offspring to 4 8 0 7, second offspring to 5 8 7 5, and the third offspring to 6 0 8 4. Replacing the weaker parents with the stronger offspring produces the results as shown in Table 3.

Step 4d. Compute elites (local best) and global best:

Global best (g_{best}) candidate solution after one generation is: "6 8 9 7" with fitness values of 30 ($6 + 8 + 9 + 7 = 30$). The computation of elite (local best) is for better understanding of the algorithm only. For better code efficiency we can directly compute global best from the evolved generations. However, for algorithm 2, which is based upon algorithm 1, computation of elites (local best of each population) is essential (Table 4).

Step 5 After a predetermined number of generations, with a given probability we replace *i*th gene of every individual with a gene whose value is the average value of the *i*th gene of individual and *i*th gene of g_{best}, i.e.,

$$\text{individual}(i) \leftarrow (\text{individual}(i) + g_{best}(i))/2$$

Table 5 summarizes the results of first iteration of computer implementation of example using the P3PGA algorithm.

Continuing further execution of the program we find that our algorithm reached the best result in about 15 iterations. Figure 3 shows number of generations (iterations) versus fitness for our example.

3 Simulated Performance, Results, and Discussion

We implemented the proposed P3PGA algorithm in MATLAB on a Core i7 @ 2.2GHz based laptop with 8GB RAM and tested its performance on 30 functions from the CEC-2014 test bench. To evaluate the performance of P3PGA we used

Table 5 First iteration and corresponding output of each major step of computer implementation of P3PGA for the example under consideration

Population no.	#1				#2				#3			
Initial	6	6	4	4	1	5	9	1	7	4	2	2
2_Parent (2P)	5	2	8	2	2	9	8	4	7	5	9	5
Population	8	4	7	5	6	6	7	1	9	6	1	6
Mitochondrial	−1	−2	−1	0	1	1	0	0	−1	−1	−1	1
Change	2	1	−1	−1	0	−1	0	1	−2	1	1	0
	0	−1	−2	−2	−1	0	1	0	−1	0	1	−1
3-parent (3P)	5	4	3	4	2	6	9	1	6	3	1	3
Population	7	3	7	1	2	8	8	5	5	6	9	5
	8	3	5	3	5	6	8	1	8	6	2	5
Best *N* individuals	8	4	7	5	2	9	8	4	7	5	9	5
Selected from 2P & 3P	6	6	4	4	2	8	8	5	5	6	9	5
In descending order	8	3	5	3	6	6	7	1	9	6	1	6
Selection of parents	8	3	5	3	2	9	8	4	5	6	9	5
For recombination	6	6	4	4	6	6	7	1	9	6	1	6
	8	3	5	3	6	6	7	1	9	6	1	6
Population	8	3	4	4	2	9	8	1	5	6	9	6
After crossover	6	6	5	3	6	6	7	4	9	6	1	5
	8	3	5	3	6	6	7	1	9	6	1	6
Population	8.0	3.0	5.0	4.0	2.0	9.0	8.0	1.0	5.0	6.0	9.0	5.0
After mutation	6.0	5.8	4.1	3.0	6.1	6.0	7.0	3.9	9.0	6.0	2.1	6.0
(genes to be rounded off)	8.0	3.0	5.0	3.0	6.0	6.0	7.0	1.0	7.8	6.0	1.0	6.0
Population after replacing weak	8	4	7	5	2	9	8	4	7	5	9	5
parents with strong	8	3	5	4	6	6	7	4	5	6	9	5
Offspring	6	6	4	4	2	8	8	5	9	6	2	6

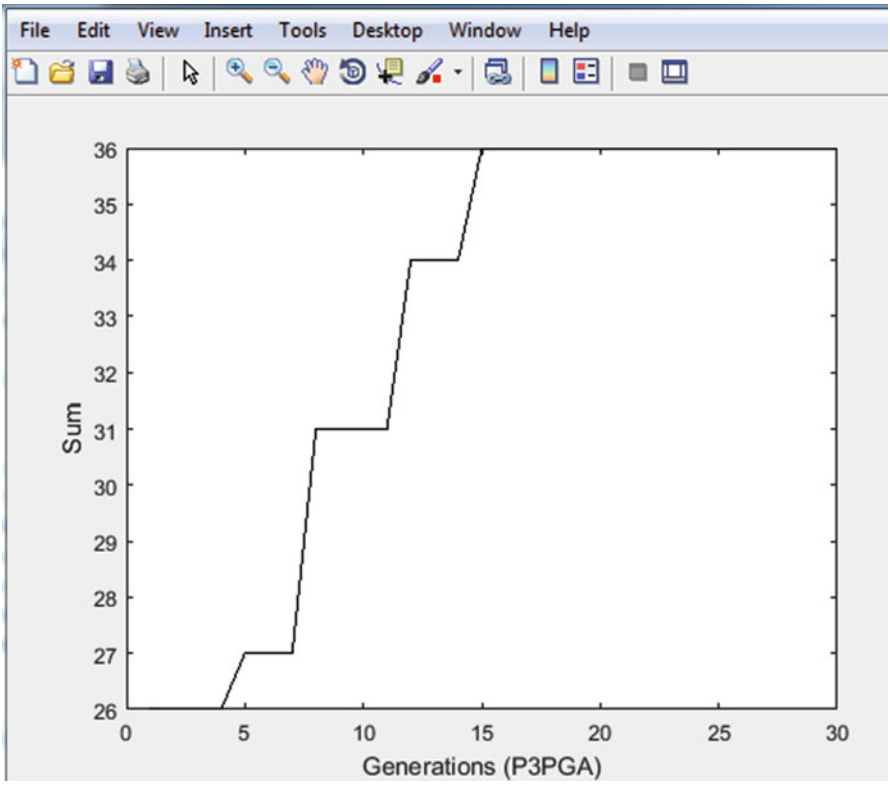

Fig. 3 Generation number versus fitness of globally best result for simple P3PGA example

$N = 10$ populations and NC $= 20$ candidate solutions. Our P3PGA results were compared to results obtained from CEC-2014 for 16 other algorithms: the United Multi-Operator Evolutionary Algorithms (UMOEAS) [20], LSHADE [21], Differential Evolution with Replacement Strategy (RSDE) [22], Memetic Differential Evolution Based on Fitness Euclidean-Distance Ratio (FERDE) [23], Partial Opposition-Based Adaptive Differential Evolution Algorithms (POBL_ADE) [24], Differential Evolution strategy based on the Constraint of Fitness values classification (FCDE) [25], Mean-Variance Mapping Optimization (MVMO) [26], RMA-LSCh-CMA [27], Bee-Inspired Algorithm for Optimization (OptBees) [28], Simultaneous Optimistic Optimization (SOO) [29], SOO+ Bound Optimization BY Quadratic Approximation (SOO + BOBYQA) [29], Fireworks Algorithm with Differential Mutation (FWA-DM) [30], algorithm Based on Covariance Matrix Leaning and Searching Preference (CMLSP) [31], Gaussian Adaptation Based Parameter Adaptation for Differential Evolution (GaAPADE) [32], Non-Uniform Mapping in Real-Coded Genetic Algorithms (NRGA) [33], and DE_b6e6rlwithrestart [34]. The

MATLAB code for the compared algorithms and the link to algorithm performance results may be obtained from http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2014/CEC2014.htm. For P3PGA we conducted 20 trials for each of the 30 functions, and took the mean error of all 20 trials as the performance measure.

The performance of P3PGA along with 16 other algorithms is given in Table 6 (see Appendix 1). Results of the comparison between algorithms are summarized in Table 7. P3PGA gave the unmatched best performance for 12 functions (f5, f9, f10, f11, f14, f15, f16, f19, f21, f24, f25, and f27) (Table 8). For f3 the performance of P3PGA was equaled by UMOEAS, RSDE, FCDE, DE_ b6e6rlwithrestart, LSHADE, MVMO, FWA-DM, and GaAPADE algorithm; and for f8 P3PGA's performance was matched by UMOEAS, FERDE, DE_b6e6rlwithrestart, GaAPADE, LSHADE, MVMO, OptBees, and RMA-LSCh-CMA. Altogether, P3PGA was the top-ranked algorithm, while UMOEAS was second.

4 P3PGA for Minimal Cost Route Evaluation

A wireless mesh network (WMN) can be mathematically represented as a set of "nodes" or points in the two-dimensional plane. These nodes represent the positions of clients, routers, and gateways that receive and retransmit communications signals. Naturally, the devices represented by nodes all have limited communication range.

Table 7 Comparative performance of P3PGA on CEC-2014 benchmarks

Algorithm	Best (unmatched)	Best (matched)	Total best	Rank
P3PGA	12	2	14	1
UMOEAS	1	8	9	2
LSHADE	2	4	6	3
DE_b6e6rlwithrestart	1	5	6	3
GaAPADE	1	4	5	4
SOO + BOBYQA	0	5	5	4
MVMO	1	3	4	5
RMA-LSCh-CMA	0	4	4	5
SOO	0	4	4	5
FCDE	0	3	3	6
RSDE	0	3	3	6
CMLSP	0	3	3	6
FERDE	0	2	2	7
POBL_ADE	1	0	1	8
FWA-DM	0	1	1	8
OptBees	0	1	1	8
NRGA	0	0	0	–

Table 8 Number of functions for which P3PGA gave the best performance

Functions for which P3PGA was a clear winner	f5, f9, f10, f11, f14, f15, f16, f19, f21, f24, f25, f27
Function for which P3PGA was a joint winner	f3, f8

In order to send a signal from a source node to a destination node, a *route* (or *path*) consisting of intermediate nodes must be found such that the signal from the source node can be successively received and retransmitted until it reaches the destination node. The process of determining the end-to-end route between a source node and a destination node is referred to as "routing." An optimal route will be one that minimizes cost, where cost is defined in terms of a routing metric. There are many possible routing metrics that appear in the literature, including minimum hop count, per hop Round Trip Time (RTT) [35], Per-Hop Packet Pair Delay (PktPair) [36], Expected Transmission Count (ETX) [37], Expected Transmission Time (ETT), Weighted Cumulative ETT (WCETT) [38], Expected Transmission on a Path (ETOP) [39], Effective Number of Transmission (ENT) and Modified Expected Number of Transmissions (mETX) [40], Metric of Interference and Channel Switching (MIC) [41], Bottleneck Link Capacity (BLC) path metric [42], cross layer link quality and congestion aware (LQCA) metric [43], and interference aware low overhead routing metric [44]. In this chapter, we use an integrated link cost function to evaluate route cost: for details see [45].

The adapted P3PGA algorithm that was used for finding optimal routes for WMNs is outlined in Algorithm 2 below. The algorithm follows all the steps of the general P3PGA algorithm described in Algorithm 1 in the previous section. These steps are described in more detail in the following paragraphs.

The first step in the algorithm is to determine initial populations of possible routes. This is done by means of an *adjacency matrix*, which is a (number of nodes) by (number of nodes) square matrix of 0's and 1's. The (i,j)th entry of the matrix is 1 if nodes i and j have a possible connection, and 0 otherwise. Using the adjacency matrix a set of route populations is generated, where each population has the same number of routes. These are the initial 2-parent populations. Next, we generate a 3-parent population from each of the current 2-parent populations and combine these two populations as in Algorithm 1. The new population is evolved from the old population using the crossover approach as given in [46]. Each 3-parent population is obtained by applying the following rules to all routes in the 2-parent population:

(a) Beginning with second node, check the location of the ND^{th} node of the local elite in the current path, where ND is defined so that nodes 2 ... ND-1 of the local elite are not in the current path, but ND is in the current path.
(b) If the ND^{th} node lies in first half of the current path, then follow the steps as given below:

I. Retain nodes of current individual up to ND and call it partial route1.
II. From the elite extract all nodes from the node after "ND" to the terminal node and call it partial route2.
III. Combine partial route1 and partial route2 to form a new path.

(c) If the "NDth" node of the local elite lies in the second half of the current path, then follow the steps as given below:

I. Retain the local elite up to "ND" and call it partial route1.
II. From the current path extract all nodes from the node after "ND" to the terminal node and call it partial route2.
III. Combine partial route1 and partial route2 to form a new route.

For example, in a WMN, let node number 1 represent the source node and node number 10 represent the terminal node. Suppose that the following two-parent population of routes exists between source and terminal node:

2-parent population 1 : [1 4 5 8 9 10; 1 3 2 7 12 8 11 9 10; 1 6 7 4 13 12 5 9 10]

In this population the local elite (shortest path) is: 1 4 5 8 9 10

Since the local elite is the fittest route (minimal cost path as per hop count method) we would not apply mitochondrial change to this elite route.

In the second route (2-parent route 2) we find that node 8 is the first node in the route that is shared with the local elite. This node lies in the second half of 2-parent route 2. Hence, we would retain the local elite up to node 8 and denote it as partial route 1:

Partial route 1 : [1 4 5 8]

From the current route (2-parent route 2), we extract all nodes starting from the node after 8 up to the terminal node, and denote it as partial route 2:

Partial route 2 : [11 9 10]

After combining partial route 1 and partial route 2 we get a new 3-parent route:

3-parent route 1 : [1 4 5 8 11 9 10]

Similarly, we make the mitochondrial change in 2-parent route 3 as follows. We first identify that the first node in the route that is shared with the elite is node 4, which lies in the first half of the current route. So we retain current route up to node 4 and denote it as partial route 1:

Partial route 1 : [1 6 7 4]

From the elite we extract all nodes after node 4 up to the terminal node, and denote it as partial route 2.

Partial route 2 : [5 8 9 10]

After combining partial route 1 and partial route 2 we obtain a second 3-parent route as follows:

3-parent route 2 : [1 6 7 4 5 8 9 10]

We combine the 2-parent routes with the newly evolved 3-parent routes into a single combined population:

[1 4 5 8 9 10; 1 3 2 7 12 8 11 9 10; 1 6 7 4 13 12 5 9 10; 1 4 5 11 9 10;
1 6 7 4 5 8 9 10]

The algorithm then evaluates the fitness of all routes in this combined population, dropping the weaker individuals and retaining the fittest N individuals, thus maintaining a constant population size.

Once this is completed, the standard genetic processes of crossover and mutation can be performed just as demonstrated in Algorithm 1, to obtain optimal solutions for all populations. The optimum of these local optima gives the current global optimum, which may then be used to update the routing table entry for the given source and destination node, so that subsequent data transfer between these two nodes may take place on the minimal cost routes. Being parallel in nature the convergence time of this algorithm is expected to be quite small.

5 Implementation and Performance of the Proposed Approach

To evaluate the comparative performance of the proposed P3PGA-based minimal cost route evaluation approach for WMNs, we implemented all the approaches in MATLAB and simulated for 100, 500, 1000, 2000, and 2500 node client WMNs. Parameters for the different WMNs are shown in Table 9. For each WMN, node locations were randomly generated within the specified area. To evaluate the performance of all approaches on 100, 500, 1000, and 2000 node client WMNs we conducted 10 trial sets, one set for a given timing constraint. On 2500 node client WMNs we considered 17 trial sets. For all networks, each trial set consists of 20 trials: to test algorithm performance in a dynamic environment, the node locations were randomly regenerated for each trial. In total, 1340 trials were conducted.

Algorithm 2: P3PGA approach for dynamic optimal cost route evaluation

Begin

/* Adjacency matrix: matrix of neighbor nodes of each node */
/* Variables:
pop_mat: Populations of routes;
path_mat: Matix whose rows are the routes in a given population;
NP: Number of populations,
N: Number of routes per population
Path_nodes: Number of nodes in a route.
*/
Calculate N routes using adjacency matrix.
Evaluate fitness of all routes in all populations.
Determine the local best routes $\ell_{best}(i)$, i = 1 . . . NP for all populations i.
Record the global best (g_{best}) route from amongst all the local best routes;
for Gen = 1: Number of Generations **do**

for pop = 1: NP **do**

/* 3 Parent Population generation starts */
3P_path = pop_mat(pop).path_mat

local_elite = ℓ_{best}(pop)
for i = 1:N **do**
SE = number of nodes in local_elite

if 3P_path(i) ! = local_elite then

for j = 2:SE-1 **do**

LOE = location of local_elite(j) in 3P_Path(i).
if LOE > 0
if LOE > path_mid then /* path_mid = half of i^{th} path*/
partial_route1 = first j nodes of local_elite
partial_route2 = nodes from LOE+1 to target node in 3P_path(i)
else
partial_route1 = first LOE nodes of 3P_path(i)
partial_route2 = nodes from j + 1 to target node in local_elite
end if
new_3P_path = concatenation of partial_route1 and partial_route2
Append new_3P_path to 3P_path
Break

end if

end for (j)

end if

end for (i) /* 3 Parent Population generation Ends */

/* Generation of 2-parent population from 3-parent population starts */
Evaluate fitness of all paths in 3P_path, sort from best to worst and select the N best paths
Select the fit paths for recombining/breeding;
With high probability recombine parents/perform cross-over.
With low probability mutate paths.
Evaluate fitness and select local_best.
Replace weak paths by stronger paths keeping the path_mat size fixed at N;
pop_mat(pop).path_mat = 3P_Path;
end for (pop) /*Generation of 2 Parent Population from 3 Parent Population Ends */
From amongst the NP local best candidates select the global best candidate g_{best};

/*move all routes of all populations towards global best*/
for i = 1:NP **do**
for j = 1:N **do**
for k = 1:Path_nodes(j) **do**
With a given probability replace k^{th} node of j^{th} route with a node of g_{best} using combination operation.
end for (k)
end for (j)
end for (i)
end for (gen)

Table 9 Architectural details of client WMNs used in simulations

No. of nodes	Area (m^2)	Radio range	Timing constraint (in seconds)
100	500 × 500	150	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0
500	500 × 500	150	0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0
1000	1000 × 1000	250	0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0
2000	2000 × 2000	250	1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5
2500	2000 × 2000	250	2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 5.5, 6.0, 6.5, 7.0, 7.5, 8.0, 8.5, 9.0, 9.5, 10.0

5.1 Comparative Performance of 100 Node Client WMNs

For 100 node client WMNs we evaluated the performance of nine approaches. The unmatched performances of all the nine approaches are shown in Table 10 and Fig. 4. Table 10 shows the total performance (matched and unmatched) of nine approaches. From the results, we observe that ACO and DSR are unreliable protocols for the given network scenarios because most of the time these protocols failed to discover any feasible routes between source-terminal pair. Being a proactive approach, the BAT approach successfully discovered feasible routes but failed to produce an unmatched optimal cost route in any of the trials.

Table 10 shows that for the timing constraint of 0.1 second, AODV produced a minimum cost route 7 + 5 = 12 times, where the first operand (7) indicates that 7 times AODV generated an unmatched optimal cost route, and the second operand (5) indicates that 5 times other algorithms obtained the same minimum cost (in this case, the other algorithms are GA, BBBC, FA, and P3PGA). The second and third place algorithms were BBBC (5 + 5 = 10 times) and P3PGA (3 + 5 = 8 times).

For the timing limit of 0.2 second, P3PGA produced minimum cost route 8 + 2 = 10 times, AODV 6 + 2 = 8 times, BBBC 3 + 2 = 5 times, and BBO produced minimum cost route 1 + 2 = 3 times. Figure 4 shows that for 100 node networks and for timing constraints less than 0.5 s (except 0.2 s) AODV performs better than all other algorithms. For timing constraint of 0.5 and 0.6 s the performance of GA is best. For timing constraint of 0.7 s BBBC and P3PGA

Table 10 Comparative performance of P3PGA on 100 node client WMNs

	Timing constraints									
Algorithm	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
AODV	7 + 5	6 + 2	5 + 9	5 + 5	1 + 9	2 + 10	2 + 11	2 + 10	4 + 0	3 + 9
DSR	16 FAIL	17 FAIL	7 FAIL	15 FAIL	1 + 4 FAIL	10 FAIL	FAIL	4 FAIL +2	12 FAIL	9 FAIL
ACO	19 FAIL	12 FAIL	16 FAIL	13 FAIL	8 FAIL	2 FAIL	5 FAIL	3 FAIL	5 FAIL	11 FAIL
GA	0 + 5	0 + 2	0 + 9	2 + 5	3 + 12	4 + 10	1 + 11	1 + 10	1 + 1	2 + 9
BBO	0 + 1	1 + 2	0 + 2	0 + 5	1 + 0	0 + 10	1 + 0	0 + 10	1 + 0	0 + 9
BBBC	5 + 5	3 + 2	2 + 9	4 + 5	1 + 9	1 + 10	3 + 11	1 + 10	3 + 1	1 + 9
FA	0 + 5	0 + 1	0 + 9	1 + 1	1 + 9	1 + 10	0 + 9	0 + 10	2 + 1	0 + 9
P3PGA	3 + 5	8 + 2	4 + 9	3 + 5	2 + 12	2 + 10	3 + 11	4 + 10	7 + 1	5 + 9
BAT	0 + 1	0 + 0	0 + 2	0 + 0	1 + 0	0 + 10	0 + 5	0 + 10	0 + 3	0 + 6

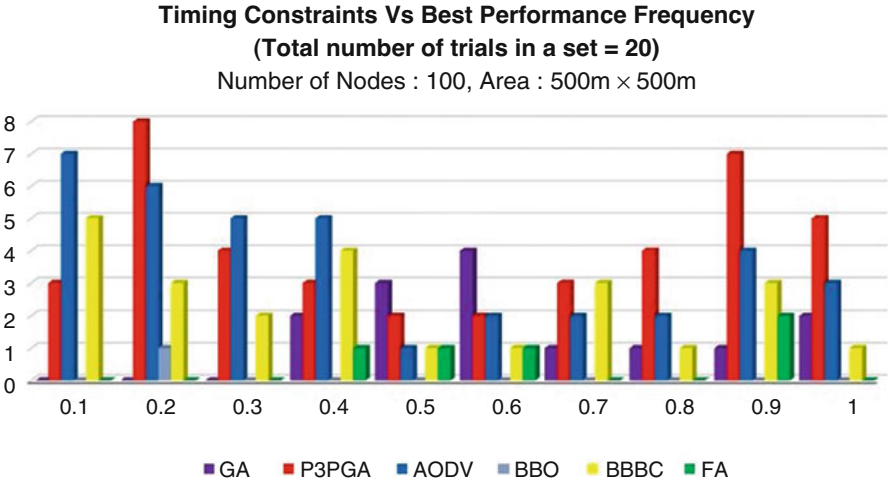

Fig. 4 Comparative unmatched best performance of 100 node client WMNs

both give best performance. As the timing constraint is further increased, more computing time is allocated to the P3PGA algorithm, so that results improve with increasing timing constraint. In terms of producing optimal cost routes, for timing constraints of 0.2, 0.8, 0.9, and 1.0 s, P3PGA algorithm outscores all other algorithms.

5.2 Comparative Performance of 500 Node Client WMNs

For 500 node client WMNs we evaluated the performance of all given 9 optimal route evaluation approaches. The performance results of the all approaches are given in Table 11 and Fig. 5. Figure 5 shows the unmatched performance of all nine approaches and Table 11 shows the total performance of all considered approaches. From results we observe that on the given WMN scenario up to 3.5 s timing constraints the AODV routing protocol outperforms all its competitors. But after

Table 11 Comparative performance of P3PGA on 500 node client WMNs

	Timing constraints									
Algorithm	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
AODV	16	10+ 1FAIL+ 5	15	14	9	19	11	6	10	4
DSR	–	–	–	–	–	–	–	–	–	–
ACO	–	–	–	–	–	–	–	–	–	–
GA	0	0 + 5	1 + 1	0	4 + 1	0	0	1 + 2	0	2
BBO	0	0 + 5	0	0	0	0	0	1 + 2	0	1
BBBC	0	1 + 5	1	2	4	1	2	1 + 2	0	3
FA	0	1 + 5	0	1	0	0	1	0 + 2	0	0
BAT	0	0 + 5	0	0	0	0	0	0 + 2	0	0
P3PGA	4	3 + 5	2 + 1	3	2 + 1	0	6	9 + 2	10	10

"–" means failed to produce route in any of the trials

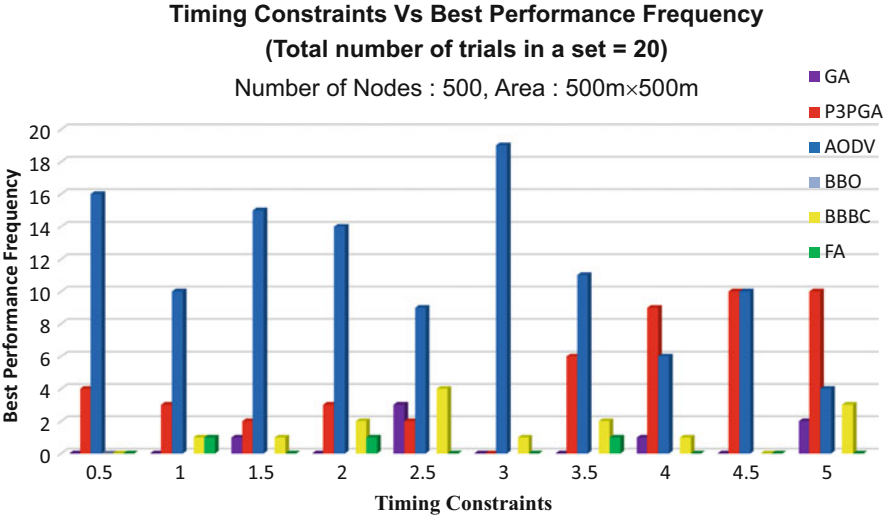

Fig. 5 Comparative unmatched best performance of 500 node client WMNs for different timing constraints

3.5 s all other approaches also started to perform. On the timing constraint of 4.0 s P3PGA produced minimum cost route $9 + 2 = 11$ times, AODV 6 times, BBBC $1 + 2 = 3$ times, Firefly $1 + 2 = 3$ times, and GA produced minimum cost route $1 + 2 = 3$ times only. With 5 s timing limit P3PGA generated minimum cost route 10 times, AODV 4 times, GA 2 times, BBO 1 time, and BBBC produced minimum cost route 3 times.

5.3 Comparative Performance of 1000 Node Client WMNs

Table 12 and Fig. 6 present the simulated performance for 1000 node client WMNs. From the performance we observe that DSR and ACO approaches failed to discover the route for the given timing constraint in any of the trial sets. Up to 2 s timing limits AODV also failed to discover any of the routes. As shown in Fig. 6, P3PGA outperforms other 8 approaches for the timing constraints of 0.5, 1.0, 1.5, 2.5, and 3.0 s. With timing constraint of 2.0 s P3PGA and BBBC gave the same best performance. Further, we also observed that after the 3.0 s timing constraint the performance of AODV improved considerably to the extent that it outperformed all other 8 approaches. Hence, for the given WMN scenarios AODV is unsuitable approach if the network size is 1000 node with allowable computing time less than 2 s. If timing constraint could be relaxed beyond 3 s, then the AODV gives the best performance.

5.4 Comparative Performance of 2000 Node Client WMNs

We simulated the performance of all the nine approaches on the timing constraints of 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, and 5.5 s. The performance results

Table 12 Comparative performance of P3PGA on 1000 node client WMNs

	Timing constraints									
Algorithm	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4	4.5	5
AODV	–	–	–	–	5	6	8	10	14	19
DSR	–	–	–	–	–	–	–	–	–	–
ACO	–	–	–	–	–	–	–	–	–	–
GA	3	2	3	2	2	1	2	0	1	0
BBO	0	0	0	0	0	0	0	0	0	0
BBBC	7	8	6	8	4	3	3	4	1	0
FA	1	0	4	2	1	1	0	0	1	0
BAT	0	0	0	0	0	0	0	0	0	0
P3PGA	9	10	7	8	8	9	7	6	3	1

"–" means failed to produce route in any of the trials

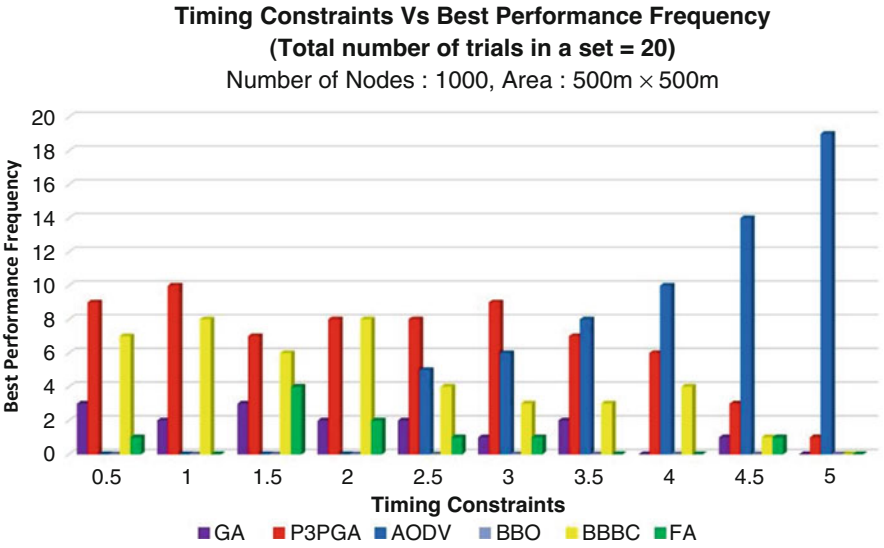

Fig. 6 Comparative performance of 1000 node client WMNs

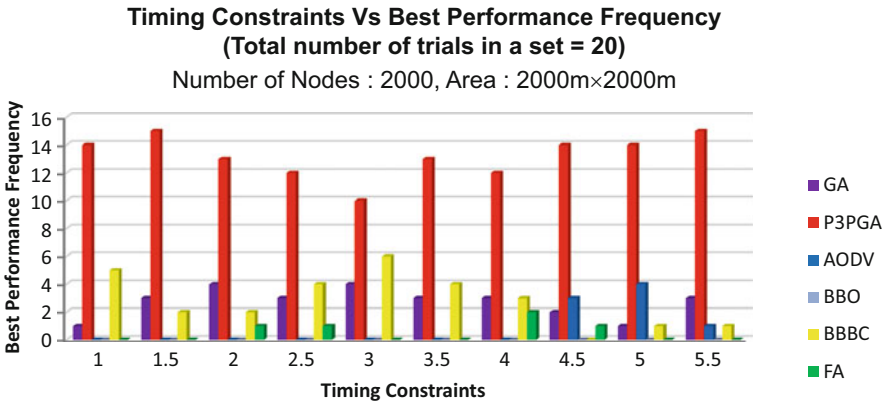

Fig. 7 Comparative unmatched best performance of 2000 node client WMNs

of all approaches are shown in Fig. 7 and Table 13. The results clearly indicate the supremacy of P3PGA approach over all other approaches for every timing constraint.

Table 13 Comparative performance of P3PGA on 2000 node client WMNs

	Timing constraints									
Algorithm	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5
AODV	–	–	–	–	–	–	0	3	4	1
DSR	–	–	–	–	–	–	–	–	–	–
ACO	–	–	–	–	–	–	–	–	–	–
GA	1	3	4	3	4	3	3	2	1	3
BBO	0	0	0	0	0	0	0	0	0	0
BBBC	5	2	2	4	6	4	3	0	1	1
FA	0	0	1	1	0	0	2	1	0	0
BAT	0	0	0	0	0	0	0	0	0	0
P3PGA	14	15	13	12	10	13	12	14	14	15

"–" means failed to produce route in any of the trials

Table 14 Comparative performance of P3PGA on 2500 node client WMNs

	Timing constraints																
Algo	2	2.5	3	3.5	4.	4.5	5.	5.5	6.	6.5	7.	7.5	8.	8.5	9.	9.5	10
AODV	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
DSR	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
ACO	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
GA	3	7	6	8	3	5	6	6	2	4	3	3	4	2	3	2	2
BBO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
BBBC	5	3	4	1	3	3	3	4	5	3	5	4	6	6	2	4	4
FA	0	1	2	1	0	2	0	1	1	1	1	2	2	1	0	1	2
BAT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P3PGA	12	9	8	10	14	10	11	9	12	12	11	11	8	11	15	13	12

"–" means failed to produce route in any of the trials

5.5 Comparative Performance of 2500 Node Client WMNs

We also evaluated the performance of all nine approaches on 2500 node client WMNs. To test the performance of all approaches we considered 17 trial sets with each set consisting of 20 trials. Here we have considered more trial sets as compared to the previous network scenarios because the network is larger and here is the need to evaluate the performance of the network on larger timing constraints also. The simulation results of all approaches are shown in Table 14 and Fig. 8. From the results we observe that the P3PGA approach outperforms all other approaches on all the timing constraints. We also observed that AODV, DSR, and ACO approaches fail to discover the route in any of the trial set. Table 15 shows that out of 340 trials, P3PGA has given the best unmatched performance 188 times, BBBC 65 times, and GA produced the best performance 69 times.

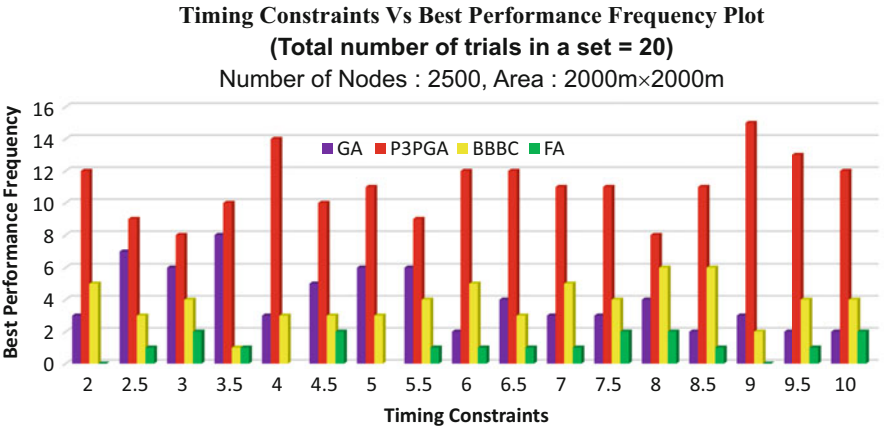

Fig. 8 Comparative performance of 2500 node client WMNs

Table 15 Overall comparative performance of P3PGA

Number of nodes	Timing constraints										
	Trials	P3PGA	FA	BBBC	BAT	AODV	BBO	GA	DSR	ACO	ALL EQUALS
100	200	41	5	24	0	37	1	14	2	0	76
500	200	49	3	15	0	114	2	7	0	0	10
1000	200	68	10	44	0	62	0	16	0	0	0
2000	200	132	5	28	0	8	0	27	0	0	0
2500	340	188	18	65	0	0	0	69	0	0	0
Total	1140	478	41	176	0	221	3	133	2	0	86

5.6 Overall Performance Considering all Networks

In order to evaluate the performance of all 9 approaches, overall we conducted total of 1140 trials. The overall performance of the 9 approaches are given in Fig. 9 and Table 15. From the simulation results, we observe that out of total number of 1140 trials P3PGA provided the unmatched best optimal cost route 478 times, AODV 221 times, BBBC 176 times, GA 133 times, Firefly 41 times, BBO 3 times, and DSR produced optimal cost routes only 2 times. 86 times multiple approaches produced the same best performance. Also, the ACO and BAT approaches failed to produce the optimal cost route in any of the trial sets. Figure 9 shows that as the size of the WMN becomes 1000 node P3PGA algorithm gives best performance but the margin is small. As the WMN size increases to 2000 nodes and above, P3PGA gives the best performance with a very large performance lead over its counterparts.

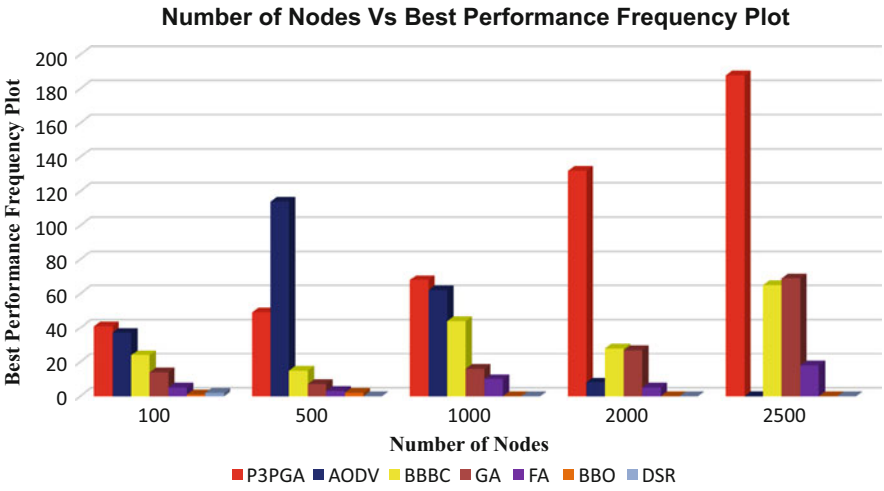

Fig. 9 Comparative performance of all approaches

6 Conclusions

This chapter proposes a new nature inspired, P3PGA-based multi-population global optimization algorithm. The proposed algorithm extended the 3PGA approach by adding the parallel evolution behavior. We implemented the proposed algorithm in MATLAB, simulated its performance on 30 benchmark functions from CEC-2014, and compared its performance with 16 other algorithms. P3PGA gave the best unmatched performance for 12 functions out of the 30 benchmark functions. On two other functions the best performance of P3PGA was equaled by some of the other algorithms. Hence, overall out of the 30 functions of CEC-2014 test suite, P3PGA gave the best performance on 14 functions. The performance of P3PGA was followed by UMOEAS, which gave unmatched best performance on one function and equaled best performance on eight functions totaling nine functions with best performance. LSHADE algorithm followed on the third place.

This chapter also proposed a P3PGA-based new optimal cost or near shortest route evaluation approach for WMNs. The approach was compared with eight other approaches, namely AODV, DSR, BBBC, ACO, BBO, BAT, GA, and Firefly-based optimal cost route evaluation approaches. From the simulation results we conclude that the proposed approach is very suitable for large WMNs with sizes greater than 1000 nodes.

The authors further suggest that the proposed P3PGA algorithm can be used in other applications such as for rule base extraction from numerical data for the fuzzy logic-based systems and for identification of fuzzy and ANN models from the given training data set.

Appendix 1: Algorithm Performance Results

Table 6 CEC-2014 Benchmark performance of various algorithms

ALGORITHM	**f1**	**f2**	**f3**	**f4**	**f5**	**f6**	**f7**	**f8**	**f9**	**f10**
NRGA	2.790E+04	9.147E+02	1.517E+03	1.544E+01	1.961E+01	2.450E+00	2.030E–01	5.585E+00	8.694E+00	1.194E+02
FWA-DM	5.013E+03	1.342E–04	0.000E+00	1.413E+00	2.003E+01	7.063E–01	9.480E–02	2.536E–01	6.008E+00	1.593E+00
UMOEAS	0.000E+00	0.000E+00	0.000E+00	0.000E+00	1.683E+01	0.000E+00	0.000E+00	0.000E+00	2.725E+00	3.739E–01
SOO + BOBYQA	4.570E+03	3.600E–02	5.843E+03	0.000E+00	2.000E+01	2.000E–03	4.900E–02	1.890E+01	8.955E+00	1.304E+02
SOO	8.811E+06	6.643E+00	6.644E+03	6.780E–01	2.000E+01	2.000E–03	4.900E–02	1.890E+01	8.955E+00	1.304E+02
RSDE	0.000E+00	0.000E+00	0.000E+00	2.811E+00	1.922E+01	5.291E–02	3.550E–02	6.608E–01	8.522E+00	6.844E+01
POBL_ADE	1.620E+04	2.270E+03	5.740E–04	2.550E+01	1.910E+01	1.040E+00	1.630E–01	7.810E+00	7.630E+00	1.530E+02
FERDE	2.368E+00	6.288E–05	1.346E–03	0.000E+00	1.906E+01	8.890E–01	1.883E–02	0.000E+00	5.638E+00	3.674E–02
FCDE	0.000E+00	0.000E+00	0.000E+00	1.841E+01	2.033E+01	3.566E+00	1.961E–01	1.607E+01	2.099E+01	2.919E+02
DE_b6e6rlwithrestart	0.000E+00	0.000E+00	0.000E+00	1.125E+00	1.845E+01	0.000E+00	1.688E–02	0.000E+00	4.895E+00	1.225E–03
CMLSP	1.769E–07	0.000E+00	1.056E–04	0.000E+00	1.686E+01	6.201E–02	0.000E+00	2.071E+00	1.659E+00	1.961E+02
GaAPADE	0.000E+00	0.000E+00	0.000E+00	3.069E+01	1.968E+01	1.484E–01	3.163E–03	0.000E+00	3.379E+00	1.518E–01
OptBees	7.842E+02	9.883E–03	9.213E–01	2.691E+00	2.000E+01	3.017E+00	1.562E–01	0.000E+00	2.084E+01	2.192E+02
LSHADE	0.000E+00	0.000E+00	0.000E+00	2.941E+01	1.415E+01	1.754E–02	3.043E–03	0.000E+00	2.345E+00	8.572E–03
RMA-LSCh-CMA	0.000E+00	0.000E+00	1.025E–07	8.501E–02	1.365E+01	1.479E–04	0.000E+00	0.000E+00	3.317E+00	7.678E+00
MVMO	4.954E–04	0.000E+00	0.000E+00	9.546E+00	1.658E+01	3.445E–03	1.858E–02	0.000E+00	3.492E+00	2.137E+00
P3PGA	**1.76E+01**	**1.480E+01**	**0.000E+00**	**5.716E–02**	**0.000E+00**	**7.236E–01**	**9.855E–02**	**0.000E+00**	**1.161E+00**	**0.000E+00**

(continued)

Table 6 (continued

ALGORITHM	**f11**	**f12**	**f13**	**f14**	**f15**	**f16**	**f17**	**f18**	**f19**	**f20**
NRGA	5.759E+02	1.242E–01	1.577E–01	2.537E–01	1.022E+00	2.747E+00	1.607E+04	7.420E+03	2.093E+00	1.719E+03
FWA-DM	3.722E+02	4.249E–02	1.206E–01	2.139E–01	7.748E–01	1.757E+00	2.545E+02	2.516E+01	1.299E+00	1.337E+01
UMOEAS	1.440E+02	0.000E+00	9.436E–03	1.100E–01	6.667E–01	1.530E+00	8.477E+00	7.840E–01	2.000E–01	3.706E–01
SOO + BOBYQA	3.491E+02	0.000E+00	3.000E–02	1.300E–01	4.200E–01	2.520E+00	4.226E+02	3.952E+03	5.500E–01	6.925E+03
SOO	3.491E+02	0.000E+00	3.000E–02	1.300E–01	4.400E–01	2.520E+00	3.123E+06	1.293E+04	5.500E–01	9.364E+03
RSDE	2.906E+02	2.206E–01	1.277E–01	1.360E–01	9.830E–01	2.233E+00	4.770E+01	1.996E+00	1.030E+00	7.215E–01
POBLADE	2.080E+02	2.690E–01	1.310E–01	2.600E–01	7.120E–01	1.410E+00	2.570E+02	3.320E+01	2.090E+00	1.260E+01
FERDE	7.554E+01	1.227E–01	1.158E–01	9.359E–02	6.725E–01	1.530E+00	8.230 + 00	2.730E+00	5.092E–01	1.704E+00
FCDE	7.554E+01	1.227E–01	1.158E–01	9.359E–02	6.725E–01	1.530E+00	8.230E+00	2.730E+00	5.092E–01	1.704E+00
DE_b6e6rlwithrestart	1.965E+02	2.929E–01	1.281E–01	1.113E–01	8.317E–01	1.872E+00	1.398E+00	6.207E–01	1.418E–01	5.593E–02
CMLSP	1.530E+02	3.027E–02	2.725E–02	1.892E–01	8.966E–01	1.555E+00	3.127E+02	3.085E+01	1.251E+00	1.994E+01
GaAPADE	1.831E+02	1.402E–01	6.009E–02	9.424E–02	6.057E–01	1.977E+00	9.914E+00	2.230E–01	2.566E–01	4.316E–01
OptBees	3.927E+02	1.304E–01	4.162E–01	3.687E–01	2.439E+00	2.640E+00	6.844E+02	3.350E+01	9.330–01	8.958E+00
LSHADE	3.206E+01	6.817E–02	5.156E–02	8.136E–02	3.661E–01	1.241E+00	9.767E–01	2.441E–01	7.730E–02	1.849E–01
RMA-LSCh-CMA	2.013E+01	1.646E–02	3.292E–02	1.265E–01	4.715E–01	1.054E+00	7.834E+01	5.221E+00	7.661E–02	8.057E+00
MVMO	9.628E+01	4.223E–02	3.553E–02	8.906E–02	4.346E–01	1.449E+00	9.357E+00	7.826E–01	1.583E–01	3.126E–01
P3PGA	**3.555E+00**	**1.365E–06**	**3.876E–02**	**2.537E–02**	**3.294E–01**	**1.662E–01**	**4.824E+01**	**6.326E+00**	**4.892E–02**	**7.684E–02**

ALGORITHM	**f21**	**f22**	**f23**	**f24**	**f25**	**f26**	**f27**	**f28**	**f29**	**f30**
NRGA	4823.427182	37.56658082	329.4574872	130.7641476	183.6782269	100.1366	280.7775666	477.1473826	413.2909874	1727.537695
FWA-DM	9.464E+01	3.409E+01	3.295E+02	1.274E+02	1.787E+02	100.1384	3.213E+02	3.472E+02	2.117E+02	3.943E+02
UMOEAS	5.404E–01	2.448E–01	3.295E+02	1.083E+02	1.260E+02	100.0140	2.548E+01	3.129E+02	1.955E+02	2.339E+02
SOO + BOBYQA	1.940E+03	1.265E+02	2.00E+02	1.157E+02	1.391E+02	100.0500	2.000E+02	2.000E+02	2.000E+02	2.000E+02
SOO	2.469E+04	1.265E+02	2.000E+02	1.157E+02	1.452E+02	100.0500	2.000E+02	2.000E+02	2.000E+02	2.000E+02
RSDE	1.209E+00	1.165E+01	3.295E+02	1.191E+02	1.295E+02	100.1291	9.125E+01	3.869E+02	2.126E+02	5.052E+02
POBL_ADE	1.030E+02	3.000E+01	3.290E+02	1.240E+02	1.860E+02	100.0000	2.560E+02	4.230E+02	3.550E+05	6.380E+02
FERDE	8.543E+00	3.242E+00	3.295E+02	1.146E+02	1.363E+02	100.0901	3.664E+02	3.664E+02	3.182E+02	5.348E+02
FCDE	1.481E+02	2.750E+01	3.295E+02	1.369E+02	1.840E+02	100.3461	4.752E+01	4.569E+02	3.405E+04	8.667E+02
DE_b6e6rlwith restart	7.867E–01	1.541E–01	3.295E+02	1.122E+02	1.290E+02	100.1170	6.161E+01	3.634E+02	2.178E+02	4.673E+02
CMLSP	3.639E+01	8.953E+01	2.018E+02	1.099E+02	1.275E+02	100.0194	4.113E+01	2.803E+02	2.000E+02	2.164E+02
GaAPADE	5.086E–01	3.247E+00	3.295E+02	1.089E+02	1.636E+02	100.0688	8.969E+01	3.832E+02	2.223E+02	4.672E+02
OptBees	5.706E+01	1.702E+01	2.724E+02	1.374E+02	1.460E+02	100.3964	7.423E+00	3.067E+02	2.200E+02	3.892E+02
LSHADE	4.081E–01	4.410E–02	3.295E+02	1.075E+02	1.327E+02	100.0500	5.806E+01	3.808E+02	2.220E+02	4.649E+02
RMA-LSCh-CMA	4.929E+01	8.475E+00	3.295E+02	1.084E+02	1.751E+02	100.0364	1.848E+02	3.887E+02	2.271E+02	5.851E+02
MVMO	1.935E+00	2.629E–01	3.295E+02	1.092E+02	1.161E+02	100.0323	1.720E+01	3.611E+02	1.814E+02	4.917E+02
P3PGA	**0.212959783**	**0.119165579**	**329.4574747**	**106.9715459**	**113.0468507**	**100.0354**	**1.26866855**	**356.3869066**	**203.2702669**	**492.4918411**

References

1. D. Goldberg, *Genetic Algorithms in Optimization, Search and Machine Learning* (Addison-Wesley, Reading, 1989)
2. B.S. Khera, P.A.P. Singh, Comparison of genetic algorithm, particle swarm optimization and biogeography-based optimization for feature selection to classify clusters of micro calcifications. J. Inst. Eng. (India): Series B **98**(2), 189–202 (2017)
3. S. Suresh Optimized scheme for grid computations using genetic algorithms, *Proceedings of the International Conference on Internet Technologies & Applications*, Wrexham, UK, September 4–7, 2007
4. M. Melanie, S. Forrest, Genetic algorithms and artificial life. Artif. Life **1**(3), 267–289 (1994)
5. J.H. Holland, *Adaptation in Natural and Artificial Systems, Ph.D. Thesis* (University of Michigan Press, Ann Arbor, MI, 1975)
6. H. Mühlenbein and H. M. Voigt, Gene pool recombination in genetic algorithms, in Meta-Heuristics: Theory and Applications, Springer US, pp. 53–62 (1996)
7. A. Eiben, C.H. Van Kemenade, Diagonal crossover in genetic algorithms for numerical optimization. Control. Cybern. **26**(3), 447–465 (1997)
8. A. Wu, P.W.M. Tsang, T.Y. Yuen, L.F. Yeung, Affine invariant object shape matching using genetic algorithm with multi-parent orthogonal recombination and migrant principle. Appl. Soft Comput. **9**(1), 282–289 (2009)
9. A.E. Eiben, P.E. Raue, and Z. Ruttkay, Genetic algorithms with multi-parent recombination, in *International Conference on Evolutionary Computation The Third Conference on Parallel Problem Solving from Nature Jerusalem, Israel*, p. 78–87 (1994)
10. P. Amato, M. Tachibana, M. Sparman, S. Mitalipov, Three-parent in vitro fertilization: Gene replacement for the prevention of inherited mitochondrial diseases. Fertil. Steril. **101**(1), 31–35 (2014)
11. H. Fertilisation and E. Authority, (2014) Third scientific review of the safety and efficacy of methods to avoid mitochondrial disease through assisted conception: 2014 update
12. J. Hamzelou, Everything you wanted to know about '3- parent' babies. [Online] (2016). Available: https://www.newscientist.com/article/2107451-everything-you-wanted-to-know-about-3-parent-babies/
13. J. Hamzelou, Exclusive: Worlds first baby born with new 3 parent technique. [Online] (2016). Available: https://www.newscientist.com/article/2107219-exclusive-worlds-first-baby-born-with-new-3-parent-technique/
14. I.F. Akyildiz, X. Wang, W. Wang, Wireless mesh networks: A survey. Comput. Netw. **47**(4), 445–487 (2005)
15. S. Amar, *Some Nature Inspired Computing Approaches to Routing in Wireless Mesh Networks, Ph.D. Thesis* (Submitted to IKG Punjab Technical University, Jalandhar (India), 2017)
16. S. Amar, K. Shakti, S. Ajay, S.S. Walia, Three-parent GA: A global optimization algorithm. J. Mult. Valued Log. Soft Comput. **32**, 407–423 (2019)
17. T. Blickle and L. Thiele, A comparison of selection schemes used in genetic algorithms, TIK Report No. 11, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Switzerland, (1995)
18. J.E. Baker Adaptive selection methods for genetic algorithms, in *Proceedings of International Conference on Genetic Algorithms and their applications*, p. 101–111 (1985)
19. J.E. Baker, Reducing bias and inefficiency in the selection algorithm. in Proceedings of the Second International Conference on Genetic Algorithms, Vol. 206, p. 14–21 (1987)
20. S.M. Elsayed, R.A. Sarker, D.L. Essam and N.M. Hamza, Testing united multi-operator evolutionary algorithms on the CEC2014 real-parameter numerical optimization, IEEE Congress on Evolutionary Computation (CEC), IEEE, p. 1650–1657 (2014)
21. R. Tanabe and A.S. Fukunaga, (2014) Improving the search performance of SHADE using linear population size reduction, IEEE Congress on Evolutionary Computation (CEC), p. 1658–1665

22. C. Xu, H. Huang and S. Ye, A differential evolution with replacement strategy for real-parameter numerical optimization. IEEE Congress on Evolutionary Computation (CEC), p. 1617–1624 (2014)
23. B.Y. Qu, J.J. Liang, J.M. Xiao and Z.G. Shang, Memetic differential evolution based on fitness Euclidean-distance ratio, IEEE Congress on Evolutionary Computation (CEC), p. 2266–2273 (2014)
24. Z. Hu, Y. Bao and T. Xiong, Partial opposition-based adaptive differential evolution algorithms: evaluation on the CEC 2014 benchmark set for real-parameter optimization", IEEE Congress on Evolutionary Computation (CEC), pp. 2259–2265 (2014)
25. Z. Li, Z. Shang, B.Y. Qu and J.J. Liang, Differential evolution strategy based on the constraint of fitness values classification, IEEE Congress on Evolutionary Computation (CEC), p. 1454–1460 (2014)
26. I. Erlich, J.L. Rueda, S. Wildenhues and F. Shewarega, Evaluating the mean-variance mapping optimization on the IEEE-CEC 2014 test suite, IEEE Congress on Evolutionary Computation (CEC), p. 1625–1632 (2014)
27. D. Molina, B. Lacroix and F. Herrera, Influence of regions on the memetic algorithm for the CEC'2014 Special Session on real-parameter single objective optimization, IEEE Congress on Evolutionary Computation (CEC), p. 1633–1640 (2014)
28. R.D. Maia, L.N. de Castro and W.M. Caminhas, Real-parameter optimization with OptBees, IEEE Congress on Evolutionary Computation (CEC), p. 2649–2655 (2014)
29. P. Preux, R. Munos and M. Valko, Bandits attack function optimization, IEEE Congress on Evolutionary Computation (CEC) (2014)
30. C. Yu, L. Kelley, S. Zheng and Y. Tan, Fireworks algorithm with differential mutation for solving the CEC 2014 competition problems, IEEE Congress on Evolutionary Computation (CEC), p. 3238–3245 (2014)
31. L. Chen, Z. Zheng, H.L. Liu and S. Xie An evolutionary algorithm based on covariance matrix leaning and searching preference for solving CEC 2014 benchmark problems, IEEE Congress on Evolutionary Computation (CEC), p. 2672–2677 (2014)
32. R. Mallipeddi, G. Wu, M. Lee and P.N. Suganthan, Gaussian adaptation based parameter adaptation for differential evolution, IEEE Congress on Evolutionary Computation (CEC), p. 1760–1767 (2014)
33. D. Yashesh, K. Deb and S. Bandaru, Non-uniform mapping in real-coded genetic algorithms, IEEE Congress on Evolutionary Computation (CEC), p. 2237–2244 (2014)
34. R. Poláková, J. Tvrdík and P. Bujok, Controlled restart in differential evolution applied to CEC 2014 benchmark functions. IEEE Congress on Evolutionary Computation (CEC), p. 2230–2236 (2014)
35. A. Adya, P. Bahl, J. Padhye, A. Wolman, and L. Zhou, A multi radio communication protocol for IEEE 802.11 wireless networks, Proceedings of International Conference on Broadcast Networks (Broad Nets), San Jose, California, USA, October 25–29, p. 344–354 (2004)
36. R. Draves, J. Padhye, and B. Zill, Comparisons of routing metrics for static multi-hop wireless networks, Proceedings of ACM Annual Conference of the Special Interest Group on Data Communication (SIGCOMM), Portland, Oregon, USA, August 30–September 03, p. 133–144 (2004)
37. D.S.J. DeCouto, D. Aguayo, J. Bicket, R. Morris, A high throughput path metric for multihop wireless routing, Proceedings of ACM Annual International Conference on Mobile Computing and Networking (MOBICOM), San Diego, CA, USA, September 14–19, p. 134–146 (2003)
38. R. Draves, J. Padhye, and B. Zill, Routing in multi-radio, multihop wireless mesh networks, Proceedings of ACM annual International conference on mobile computing and networking (Mobi Con04), Philadelphia, Pennsylvania, USA, September 26–October 01, p. 114–128 (2004)
39. G. Jakllari, S. Eidenbenz, N. Hengartner, S. Krishnamurthy, and M. Faloutsos, Link positions matter: A noncommutative routing metric for wireless mesh networks, Proceedings of IEEE Annual Conference on Computer Communications (INFOCOM), Phoenix, Arizona, USA, April 13–18, p. 744–752 (2008)

40. C.E. Koksal, and H. Balakrishnan, Quality-aware routing metrics for time varying wireless mesh networks, IEEE Journal on Selected Areas in Communications, 24(11), p. 1984–1994 (2006)
41. Y. Yang, J. Wang, R. Kravets, Interference-aware load balancing for multi hop wireless networks, Technical Report UIUCDCSR-2005-2526, University of Illinois at Urbana Champaign, Department of Computer Science, and Web Address: http://www.ideals.uiuc.edu/handle/2142/10974, (2005)
42. T. Liu and W. Liao, Capacity-aware routing in multi-channel multi-rate wireless mesh networks, Proceedings of IEEE International Conference on Communications (ICC), Istanbul, Turkey, 11–15 June, p. 1971–1976 (2006)
43. G. Karbaschi and A. Fladenmuller, A link quality and congestion-aware cross layer metric for multi-hop wireless routing, Proceedings of IEEE International Conference on Mobile Ad hoc and Sensor Systems Conference, Washington, DC, USA, 2005, 7 Nov. 2005, p. 7–11
44. L. Ma, Q. Zhang, Y. Xiong, and W. Zhu, Interference aware metric for dense multi-hop wireless network, Proceedings of IEEE International Conference on Communications (ICC), Seoul, South Korea, pp. 1261–1265 (2005)
45. S. Sharma, S. Kumar, B. Singh, Routing in wireless mesh networks: Three new nature inspired approaches. Wirel. Pers. Commun. **83**(4), 3157–3179 (2015)
46. S. Yang, H. Cheng, F. Wang, Genetic algorithms with immigrants and memory schemes for dynamic shortest path routing problems in mobile ad hoc networks. IEEE Trans. Syst. Man Cybern. Part C Appl. Rev. **40**(1), 52–63 (2010)