A Multi-memory Multi-population Memetic Algorithm for Dynamic Shortest Path Routing in Mobile Ad-hoc Networks

Nasser R. Sabar(✉), Ayad Turky, and Andy Song

School of Computer Science and I.T., RMIT University, Melbourne, Australia
{nasser.sabar,ayad.turky,andy.song}@rmit.edu.au

Abstract. This study investigates the dynamic shortest path routing (DSPR) problem in mobile ad-hoc networks. The goal is to find the shortest possible path that connects a source node with the destination node while effectively handling dynamic changes occurring on the ad-hoc networks. The key challenge in DSPR is how to simultaneously keep track changes and search for the global optima. A multi-memory based multi-population memetic algorithm is proposed for DSPR in this paper. The proposed algorithm combines the strength of three different strategies, multi-memory, multi-population and memetic algorithm, aiming to effectively explore and exploit the search space. It divides the search space by multiple populations. The distribution of solutions in each population is kept in the associated memory. The multi-memory multi-population approach is to capture dynamic changes and maintain search diversity. The memetic component, which is a hybrid Genetic Algorithm (GA) and local search, is to find high quality solutions. The performance of the proposed algorithm is evaluated on benchmark DSPR instances under both cyclic and acyclic environments. Our method obtained better results when compared with existing methods in the literatures, showing the effectiveness of the proposed algorithm in handling dynamic optimisation.

Keywords: Dynamic shortest path routing · Memetic algorithms · Dynamic optimisation · Evolutionary algorithm

1 Introduction

This study is to establish a new method for solving the dynamic shortest path routing (DSPR) problem under mobile ad-hoc networks (MANET) environments where the topological structure of network keeps changing. MANET is made of an arbitrary group of mobile devices such as mobile phones. Nodes may be appearing or disappearing on the network due to factors like flat battery, poor reception, interrupted services and so on [12]. Unfortunately optimisation algorithms that have been proposed to solve static shortest path routing (SPR) problem for MANETs are not directly suitable for DSPR [2,11,13]. Because dealing with dynamic environments requires tracking changes and searching for

R. Booth and M.-L. Zhang (Eds.): PRICAI 2016, LNAI 9810, pp. 406–418, 2016.
DOI: 10.1007/978-3-319-42911-3_34

optimal solutions simultaneously. One remedy to this issue is through maintaining the search diversity so the search process can cope with problem changes more effectively.

A multi-memory multi-population memetic algorithm (M-MMA) is therefore proposed for DSPR. It is built upon a recently established memetic algorithm for DSPR. Memetic algorithm incorporates local search algorithm with Genetic Algorithm (GA) so local exploitation can be combined with exploration [5,10]. Our algorithm introduces two extra components, multi-memory and multi-population to further improve the search process. Multi-population is to divide a population of solutions into several sub-populations [9]. Each sub-population occupies a different region of the search space. An area which was bad but becomes good may be quickly identified by the search. The second component multi-memory is to maintain the solution distribution of each sub-population. The improvement of solutions in each sub-population is done through memetic algorithm (MA). A well-known DSPR simulator proposed by Yang et. al is used in our study [12]. Experiments show that the proposed algorithm can achieve better performance compared to state-of-the-art algorithms in the literature.

2 Problem Description

DSPR problems can be represented as an undirected connected graph in which there are a set of nodes and a set of edges, $G(V, E)$. Each node v ($v \in V$) represents a mobile device or a wireless router. All nodes are connected by edges that link adjacent nodes. Each edge is associated with a weight or a cost that represents distance or the cost of communication between v_i to v_j. On a MANET two nodes will be connected if they can reach each other for packet transmission. Hence any two nodes within the radio transmission range of each other and operating on the same channel will be connected. For each connection or edge, a transmission delay is also added. Due to the dynamic nature of MANET, the topology may change over time. An initial network G_0 may change to G_1, G_2 to G_n.

The formal notation for DSPR in MANETs is presented in Table 1. This notion is from [3]. The main goal to find the shortest possible path between the source node s and the destination node t. The generated path is considered feasible if it contains no loops (loop-free) and the total communication delay is within the upper bound. When a change occurs in a MANET, meaning devices joining or leaving the network, a DSPR algorithm should still be able to find a feasible and shortest path to reconnect. Thus an effective DSPR algorithm should response to a change very quickly regardless the nature of the change. The objective functions of DSPR can be formulated as follow:

$$D(P_i) = \sum_{l \in P_i(s,t)} d_i \leq DELAY \tag{1}$$

$$C(P_i) = \min_{P_i \in G_i} \sum_{l \in P_i(s,t)} c_l \tag{2}$$

Table 1. Notation of DSPR for MANETs

$G_0(V_0, E_0)$	A graph representing the initial MANET
$G_i(V_i, E_i)$	Graph of the MANET after the ith change
s	Source node
t	Destination node
$P_i(s, t)$	Path from node s to r on graph G_i
l	A link connecting two nodes
d_l	Transmission delay on link l
c_l	Communication cost of link l
$D(P_i)$	Total transmission delay on path P_i
$C(P_i)$	Total cost of path P_i

where $DELAY$ is the delay upper bound. The total delay $D(P_i)$ along the transmission path from s to t should not exceed delay upper bound $DELAY$ and the total cost $C(P_i)$ of the transmission should be minimum.

3 Methodology

Our proposed M-MMA method combines the strengths of three strategies: (1) multi-memory, (2) multi-population, (3) memetic algorithm. This approach aims to search for good solutions while effectively respond to dynamic changes occurred during the optimisation process. These three strategies are applied in a sequence on a given problem instance as follows. Firstly, the multi-population component divides the entire population into several sub-populations. Secondly, the solutions of each sub-population is improved by memetic algorithm through evolutionary operators including selection, crossover and mutation, and a local search process. Thirdly, the memory mechanism is called to update the solutions of each sub-population.

The flowchart of the algorithm is shown in Fig. 1. It first sets the parameters, randomly create an initial population of solutions and evaluate their fitness value. Next, it divides the population into m sub-populations. The aforementioned three strategies are then applied on each sub-population separately. Once a change in the environment is detected, all solutions are merged into one big population to be re-partitioned again. This process is repeated until the stopping criteria is met. The details are discussed in the following subsections.

3.1 Set Parameters

The proposed algorithm has six parameters: the maximum number of iterations ($MaxIt$), population size (Ps), memory size (M_s), crossover rate (CR), mutation rate (MR) and the number of sub-populations (m). The value of each parameter is set based on preliminary tests which are discussed in Sect. 4.3.

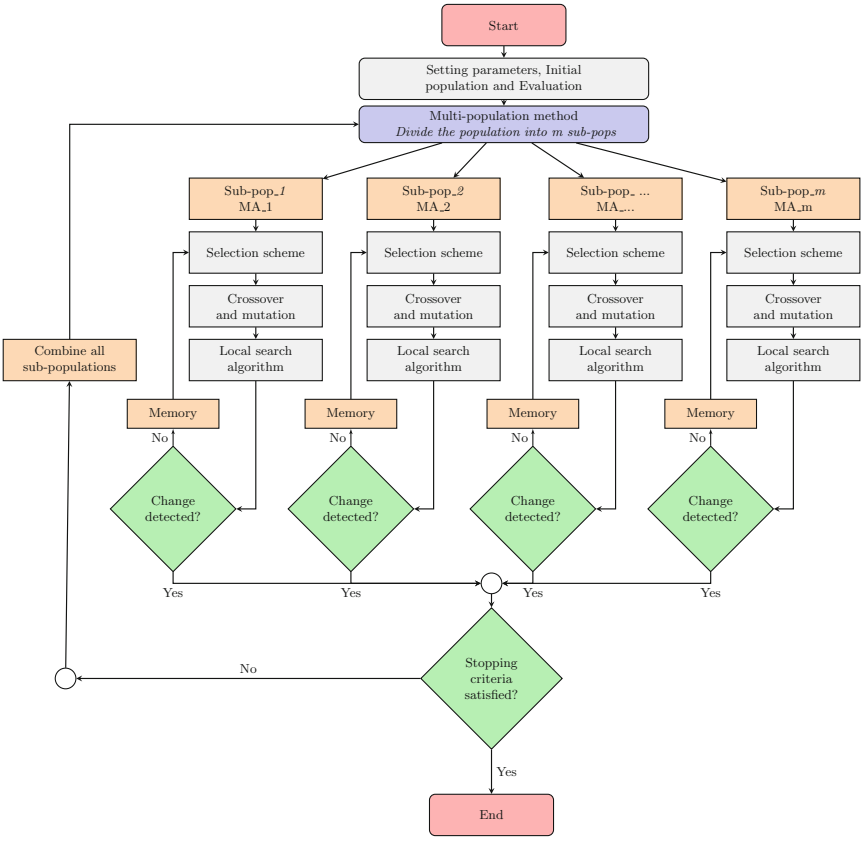

Fig. 1. Flowchart of our Multi-memory Multi-population memetic algorithm

3.2 Initial Population

The population of solutions with size Ps are randomly generated. Each solution is represented by one chromosome that is a one-dimensional array. Each gene contains an integer number which represents the ID of a node on the MANET *G(V, E)*. The first and last genes are the source and destination nodes respectively.

3.3 Evaluation

The fitness values of solutions, including the initial solutions, are calculated using the following equation:

$$f(s) = \left[\sum_{l \in p(s,t)} C_l \right]^{-1} \tag{3}$$

where $f(Ch_i)$ represents the fitness value of chromosome (Ch_i), s and t represent the source and destination nodes respectively, *p(s, t)* is the path between the source node and destination node, l is the link between nodes in *p(s, t)* and C is the cost of l^{th} link. It can be seen that low cost leads to high fitness value, hence having better chance to be selected for reproducing the next generation.

3.4 Multi-population

The multi-population component randomly divides the whole population into m sub-populations. Each sub-population is to be optimised by MA. All sub-populations interact with each other through merging and re-partitioning once a change in the environment is detected.

3.5 Memetic Algorithm

Memetic algorithm (MA) is a well-known stochastic optimisation algorithm [5,6]. It is a hybrid scheme that combines the exploration aspect of population based evolutionary search and the exploitation aspect of local search [8]. The MA in this study hybridises genetic algorithm (GA) with local search.

3.6 Genetic Algorithm

Genetic algorithm (GA) is a well-known problem solving method inspired by survival-of-the-fittest principle in nature [3]. In our method, each sub-population has its own GA process which involves the following major steps.

- **Selection**: Selection picks up two solutions from the population for reproduction [3]. A pair-wise without replacement tournament selection scheme is used here [4,12]. Solutions in a population are randomly paired. The better one from each pair will be considered as winner for that tournament.
- **Crossover**: Crossover exchanges the genes of two selected solutions to generate new solutions of the next generation [3]. Single-point crossover operator is used here [1,4,12]. In our method that single point selected as the crossover point is always an intermediate node between the source node and the destination node. Hence the new solutions generated by crossover have same source and destination. The probability of performing crossover is determined by the crossover rate CR.
- **Mutation**: Mutation complements crossover in allowing the search to get out from the local optima point [3]. A one-point mutation operator is used [1,7]. It first randomly chooses a point as the mutation point and then randomly changes the values of all points behind that mutation point.

– **Repair procedure**: Both crossover and mutation may generate infeasible solutions which contain loops in the path. The repair procedure is to fix infeasible solutions and make feasible. It removes loops by eliminating duplicated node and reconnecting the path with neighboured one [7,12].

3.7 Local Search

Local search aims to improve the convergence of the search process. Simple descent is used in this work. It iteratively explores the neighbourhood area of a given solution, seeking for a better alternative. In each iteration, a neighbourhood solution is generating by modifying the current solution using a replace operator. This operator randomly select one node and then replace it with one of its neighbouring nodes. The updated solution will be accepted if it is better than the original one in term of the fitness measure. This iterative process continues until the termination condition of the local search is met. In our M-MMA the local search will stop if there is no improvement after a predefined number of iterations (see Sect. 4.3). To reduce computational cost, the local search is only triggered if the fitness value of the new solution is no better than the worst one in the population.

3.8 Change Detection

This part checks whether there is a significant difference in the environment, meaning G_n is different with G_{n+1}. If that is positive then the search process of all sub-populations will terminate. Otherwise, a new generation will start for every sub-population.

3.9 Multi-memory

The main role of memory is to ensure that the solutions of each sub-population are well scattered over the landscape. So changes can be dealt with more promptly. At each generation, redundant solutions in the population are removed and replaced with solutions stored in the memory. In this study three different types of memories are introduced. Each type stores a set of solutions described below. Each MA process is randomly assigned with one type of memory:

1. M1: a set of random solutions.
2. M2: the best solutions from the previous generations.
3. M3: a set of solutions generated by modifying existing best solutions.

3.10 The Stopping Criteria

M-MMA terminates if the maximum number of generations is reached. Otherwise, when a change is detected all sub-populations will be combined to form a large population which will be again divided to multi-population search for the new MANET topology.

4 Experiments

This section describes our experiments including the simulation of dynamic MANET, the performance evaluation metric and the parameter settings of M-MMA.

4.1 The Simulation of Dynamic Environments

The dynamic simulator for MANET and the network topology instances are introduced by [12]. The simulator first generates a square region of 200×200, where the x axis coordinate and y axis coordinate are set between $[0, 200)$. Next, it randomly places 100 nodes and establish links between them. Nodes are linked if the Euclidean distance is less than the given radio transmission *Range*, where, *Range* is set to 50. Each link is randomly assigned a cost and delay. The delay upper bound is twice the minimum end-to-end delay, same as that in [12]. These steps will continue until all connections are created and the network is established.

To simulate a dynamic aspect, the initial topology will be changed over the time by modifying a number of selected nodes. Two parameters, R and M are to control the dynamic environment. R represents the number of generations between consecutive changes, while M represents the severity of change. For instance, if R is set to 5 then the topology will be changed for every 5 generations during evolution. If M is set to 2 this mean at each change 2 nodes will be randomly selected and changed. Each node must be either in active or sleep mode. If a selected node is active, its status will be changed into inactive. Similarly if a node is inactive, then it will be activated to join the network.

The simulator instances consists of four series of different characteristics. These sets are namely series #1, series #2, series #3, and series #4. Two different dynamic environments are considered in this paper: acyclic dynamic environment (series #2, #3 and #4) where there is no repeat of topology and cyclic dynamic environment (represented by series #1) where the repeat of topology is permitted. In series #2, #3 and #4, M is set to 2, 3 and 4, respectively. In series #1, M is equal to 2. Network topology 1 is set the same as the last one, topology 21.

4.2 Performance Evaluation Metric

In this paper, we use the overall off-line performance ($\overline{F}_{OFF}$) to evaluate the performance of the proposed M-MMA. This measurement is also used by others for algorithm comparisons [12]. It can be calculated as follows:

$$\overline{F}_{OFF} = \frac{1}{Max_{gen}} \sum_{i=1}^{Max_{gen}} (\frac{1}{N_{run}} \sum_{j=1}^{N_{run}} \sum_{l \in p(s,t)} C_{l,i,j}^{best}) \quad (4)$$

where Max_{gen} and N_{run} represent the maximum number of generations and the total number of runs respectively. $C_{l,i,j}^{best}$ is the cost of a link on the path of the best solution at the i^{th} generation during the j^{th} run. The lower the $\overline{F}_{OFF}$ value, the better the performance. Note this measurement is different with the objective function shown in Formula 3. The objective function is to guide the search.

4.3 Parameter Settings

The proposed algorithm has seven different parameters that need to be set by user. To tune these parameters, a series of preliminary experiments were conducted to find out the most appropriate value for each of these parameters. We tested the proposed M-MMA 30 independent runs with different parameters combinations. The best parameter values are listed in Table 2.

Table 2. The parameter settings

M-MMA Parameter	Tested values	Suggested value
Population size (PS)	10, 20, 30, 40, 50, 60, 70	30
Crossover rate (CR)	0.3, 0.5, 0.7, 0.9	0.7
Mutation rate (MR)	0.1, 0.3, 0.6 , 0.9	0.1
Number of sub-population (m)	3, 5, 7 , 10	5
Consecutive non-improvement iterations	5, 10, 15 , 20	5
Memory size	1–20	8

5 Results and Discussions

The experimental results are presented here and compared with other methods. M-MMA was tested on four series of network instances for cyclic dynamic environments and the acyclic dynamic environments.

5.1 Results Under Cyclic Dynamic Environment

M-MMA is compared with the following three algorithms taken from the literature for cyclic dynamic environment:

1. **MEGA**:Genetic algorithm with memory scheme [12].
2. **MRIGA**: Memory and random immigrants GA [12].
3. **MIGA**: Memory based immigrants GA [12].

Series #1 instances are used here. Parameter M is set to 2, while R value, time for change, is set to 5, 10, and 15 respectively. Series #1 contains 101 topologies. The maximum number of generations for *R=5*, *R=10* and *R=15* is 505, 1010 and 1515, respectively.

Table 3 shows the results of M-MMA. Results from other three methods, MEGA, MRIGA and MIGA on series #1 are also listed. The best results are highlighted in bold. Observing these results we see the superb performance of our proposed M-MMA comparing with MEGA, MRIGA and MIGA. M-MMA outperformed these algorithms on all instances.

Table 3. Results under cyclic dynamic environment

Algorithm	Series #1		
	R=5	*R=10*	*R=15*
M-MMA	**432.427**	**418.686**	**416.811**
MEGA	464.935	427.622	442.997
MRIGA	437.23	440.984	430.033
MIGA	480.446	461.834	445.168

To examine the search behaviour of our M-MMA as well as MEGA, MRIGA and MIGA, the search progress over 500 generations for R=5 is plotted in Fig. 2. As can be seen from the figure, M-MMA is consistently at the bottom of the figure. It is most stable one during the whole the search process. In comparison other methods fluctuate when a change occur. This illustrates that our M-MMA is an effective solution method for handling DSPR problem with cyclic changes.

5.2 Results Under Acyclic Dynamic Environments

In this set of experiments under acyclic dynamic environment, our M-MMA is compared with another three methods proposed for this situation. These methods are:

1. **EIGA**: Elitism based immigrants genetic algorithm [12].
2. **RIGA**: Random immigrants genetic algorithm [12].
3. **HIGA**: Hybrid immigrants genetic algorithm [12].

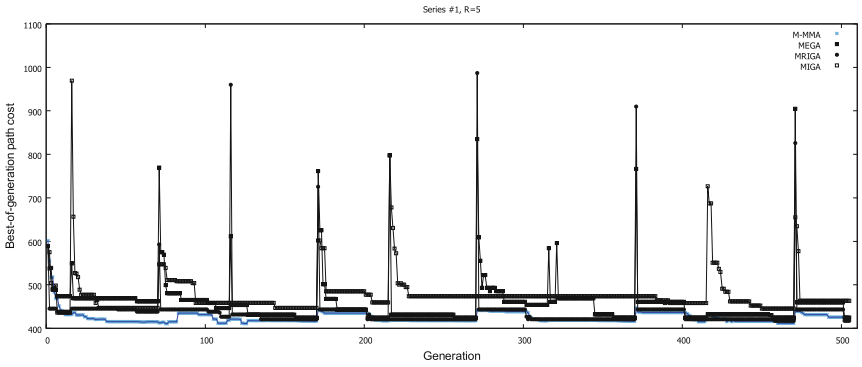

Fig. 2. Search progress of four algorithms on Series #1, $R = 5$

Series #2, Series #3, and Series #4 are used here. Each series involves 21 different network topologies. The M values are set to 2, 3 and 4. The R value are set to be 5, 10 and 15 respectively. The maximum number of generations for these three values are 105, 210 and 315, respectively. So all 21 topologies can be included in the experiments.

Table 4 shows the results from these four methods. The best average result on one row is highlighted in bold. As can be seen from the table, our M-MMA outperformed EIGA, RIGA and HIGA on all series. This good performance is mainly the result of multi-memory and multi-population which can preserve the diversity during the search process while removing redundant solutions.

The search progress of the four algorithms are plotted in Fig. 3 on Series #2, Series #3, and Series #4 when R=5. As can be seen from the figure our M-MMA again exhibits its stability under these dynamic environments. M-MMA was the lowest curve on these plots and stayed consistently low when changes occur. These changes on the network caused high path cost, or a spike on the figure, when using other three methods. This comparison shows that the proposed M-MMA can quickly adjust itself to the changes under acyclic dynamic environment.

Table 4. Results under acyclic dynamic environments

	Series #2			Series #3			Series #4		
	R=5	*R=10*	*R=15*	*R=5*	*R=10*	*R=15*	*R=5*	*R=10*	*R=15*
M-MMA	**427.142**	**421.01**	**417.643**	**460.124**	**429.75**	**424.749**	**446.732**	**434.839**	**431.69**
EIGA	447.743	446.371	436.565	462.619	485.471	448.524	489.762	468.438	462.937
RIGA	433.838	435.886	440.587	461.19	450.352	445.263	475.543	461.819	464.737
HIGA	445.381	467.776	455.333	490.962	497.276	452.422	506.962	487.052	498.724

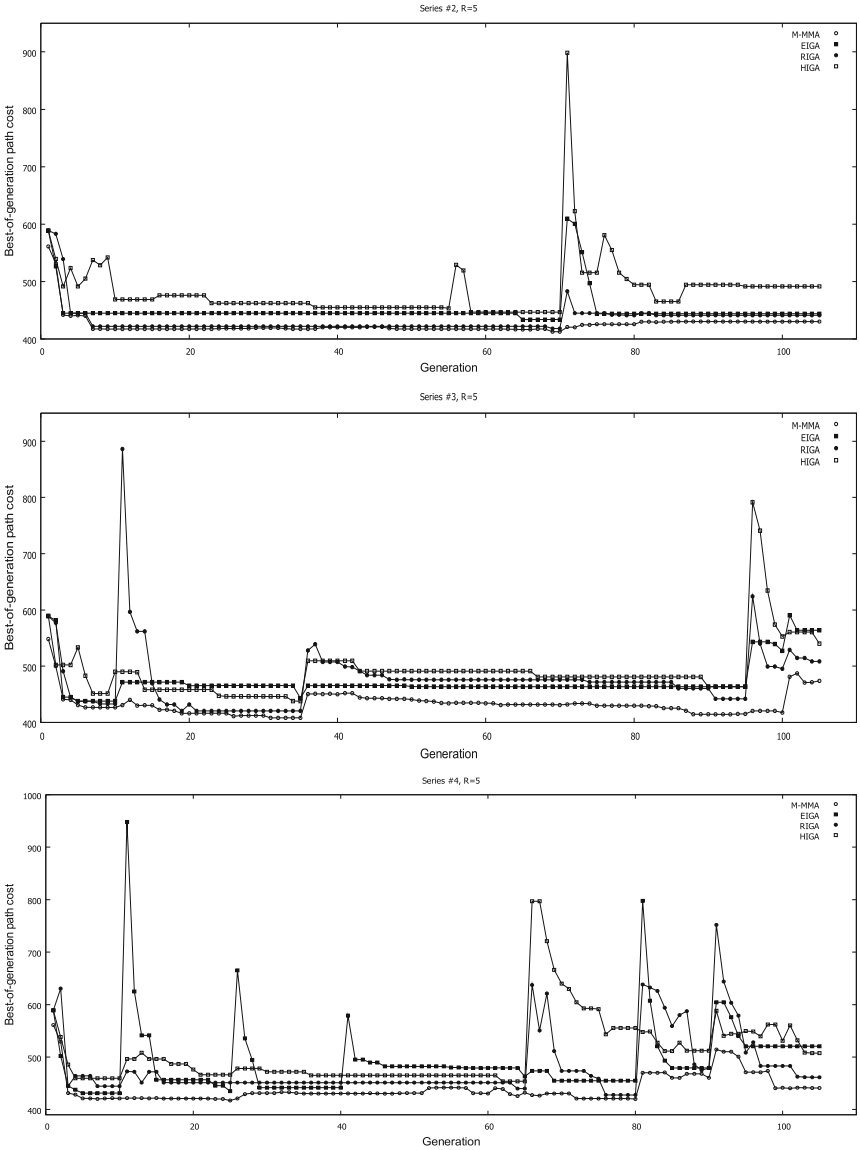

Fig. 3. Search progress of four algorithms on Series #2, #3, #4 ($R = 5$)

6 Conclusion

This study proposed a multi-memory multi-population memetic algorithm for dynamic shortest path routing problems in mobile ad-hoc networks. The proposed algorithm divides a population of solutions into several sub-populations to perform search separately over different parts of the search space. It use a multi-memory mechanism to store solutions for each sub-population so good solutions can be preserved to cope with future changes. Memetic algorithm, which hybridises genetic algorithm and local search, is performed on each sub-population to find high quality solutions for that sub-population.

The proposed method has been evaluated on four series of shortest path routing problems under different dynamic environment. Six different state-of-the-art methods were introduced for comparison. The results shown that our method can handle both cyclic and acyclic dynamic changes without modifications. More importantly the shortest paths found by the proposed method are better than paths found by other methods. Further analysis shows that the search performance of the proposed M-MMA is very stable. It can quickly adjust itself to fit with the new environment. The search is consistent yet efficient in terms of coping with dynamic changes. We conclude that the proposed multi-memory multi-population memetic algorithm is an effective and competitive approach in solving dynamic shortest path routing problems. It can accommodate changes well while performing search. It is a good candidate for dynamic MANETs.

In our future study we will examine the exact contribution of the memory and multi-population components. So the performance may be further improved, the computational cost may be reduced. In addition more instances will be introduced to facilitate further validation and extension.

References

1. Ahn, C.W., Ramakrishna, R.S.: A genetic algorithm for shortest path routing problem and the sizing of populations. IEEE Trans. Evol. Comput. **6**(6), 566–579 (2002)
2. Branke, J.: Evolutionary Optimization in Dynamic Environments, vol. 3. Springer Science & Business Media, New York (2012)
3. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT press, Cambridge (1992)
4. Lee, S., Soak, S., Kim, K., Park, H., Jeon, M.: Statistical properties analysis of real world tournament selection in genetic algorithms. Appl. Intell. **28**(2), 195–205 (2008)
5. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Caltech Concurrent Comput. Program, C3P Rep. **826**, 1989 (1989)
6. Neri, F., Cotta, C.: Memetic algorithms and memetic computing optimization: a literature review. Swarm Evol. Comput. **2**, 1–14 (2012)

7. Oh, S., Ahn, C.W., Ramakrishna, R.S.: A genetic-inspired multicast routing optimization algorithm with bandwidth and end-to-end delay constraints. In: King, I., Wang, J., Chan, L.-W., Wang, D.L. (eds.) ICONIP 2006. LNCS, vol. 4234, pp. 807–816. Springer, Heidelberg (2006)
8. Sabar, N.R., Song, A.: Dual population genetic algorithm for the cardinality constrained portfolio selection problem. In: Dick, G., Browne, W.N., Whigham, P., Zhang, M., Bui, L.T., Ishibuchi, H., Jin, Y., Li, X., Shi, Y., Singh, P., Tan, K.C., Tang, K. (eds.) SEAL 2014. LNCS, vol. 8886, pp. 703–712. Springer, Heidelberg (2014)
9. Sabar, N.R., Song, A., Tari, Z., Yi, X., Zomaya, A.: A memetic algorithm for dynamic shortest path routing on mobile ad-hoc networks. In: 2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS), pp. 60–67. IEEE (2015)
10. Sabar, N.R., Song, A., Zhang, M.: A variable local search based memetic algorithm for the load balancing problem in cloud computing. In: Squillero, G., Burelli, P. (eds.) EvoApplications 2016. LNCS, vol. 9597, pp. 267–282. Springer, Heidelberg (2016). doi:10.1007/978-3-319-31204-0_18
11. Turky, A.M., Abdullah, S., Sabar, N.R.: A hybrid harmony search algorithm for solving dynamic optimisation problems. Procedia Comput. Sci. **29**, 1926–1936 (2014)
12. Yang, S., Cheng, H., Wang, F.: 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)
13. Yang, S., Yao, X.: Population-based incremental learning with associative memory for dynamic environments. IEEE Trans. Evol. Comput. **12**(5), 542–561 (2008)