HARMONY SEARCH BASED ALGORITHM FOR DYNAMIC SHORTEST PATH PROBLEM IN MOBILE AD HOC NETWORKS

Amir I. Amin[1,*], Haitham S. Hamza[*], Imane A. Saroit[*]

[*]Faculty of Computers and Information, Cairo University, Egypt

***Abstract*—Recently, more and more attention is drawn to wireless communication especially Mobile Ad Hoc Networks (MANET) for the flexibility they provide. In MANETs mobile nodes make use of the infrastructure-less, decentralized structure to freely move without any restrictions. However, due to the limited battery life of the nodes, using the shortest path in communication is essential to conserve energy. Shortest Path (SP) problem in MANETs is a dynamic problem because of the unpredictable MANET environment and the continuously changing topology. Thus, trying to get the SP with at least two additive metrics is a NP-complete problem. In this paper, we propose a modified Harmony Search (HS) algorithm to rapidly find a high quality sub-optimal solution to the dynamic SP problem in MANETs. Our empirical results show that our modified HS can recover from topological changes and rapidly converge to good solutions before another topological change takes place.**

***Keywords*—** MANET, Harmony Search, Evolutionary Algorithms, Dynamic Optimization, Shortest Path.

1. INTRODUCTION

Mobile Ad Hoc Network (MANET) [1] is a collection of wireless mobile nodes dynamically forming a temporary network without the use of any existing network infrastructure or centralized administration. The decentralized, infrastructure-less nature creates a distributed architecture in which the mobile nodes cooperate to provide the services.

The main service/operation of networks is relaying information, which completely depends on routing. Routing in MANETs is very complex due to a lot of constraints imposed by the nature of MANETs such as mobility, limited battery life, and mutual interference of nearby nodes, among other constraints. In wired or fixed infrastructure wireless networks, Dijkstra, breadth first and Bellman-Ford shortest path (SP) techniques are used in routing protocols. These protocols have polynomial time complexity, making them unsuitable for highly dynamic environments like MANET as they would introduce large time delays because of their high computational complexity [2].

The computational complexity of SP problem with at least two additive metrics is NP-complete [3]. Thus, mathematically based optimization techniques can only be employed after the decomposition or reduction of this NP-complete problem into smaller, more manageable sub-problems [4]. On the other hand, meta-heuristic optimization techniques can be used to solve this type of problems without any decomposition of the main problem. A lot of our daily life problems are changing dynamic problems by nature, so the ability to get an optimal solution for this type of problems is important. Optimization problems changing over time are called dynamic or time dependant problems [5]. Solving dynamic optimization problems (DOP) is more challenging since it requires the optimization technique not only to find the optimal solution but also to track the changing optimal solution. As in [5] evolutionary algorithms (EA) and swarm intelligence (SI) are good candidates to solve DOPs because they are inspired from biological evolution and self organized systems, these systems if not adaptable by nature they exhibit the potential to adapt to environmental changes, which is beneficial in solving optimization problems in dynamic environments like routing in MANETs.

In stationary optimization problems the goal is to converge to the global optimal solution as quickly as possible. In EA and SI as search converges to an optimal solution the population gets more and more homogenous. After exploring the search space the optimization algorithm focuses on a promising area and starts to exploit that area seeking an optimal

[1] Corresponding Author: a.ibrahim@fci-cu.edu.eg

solution, which explains the homogeneity of the population. When a change happens causing the optimal solution to move to another area of the search space. The ability of the optimization technique to re-converge to the new optimal solution is highly dependent on the diversity level of population right after the change happened.

One possible way of finding the new optimal solution in a dynamic environment is to restart the optimization technique all over again, the problem with that technique is the long time it takes to re-converge to an optimal solution and the loss of previous search experience. In most environments whenever a change happens it affects part of the search space not the whole space, this is why not using previous search experience represents a big loss. This is where the idea of most evolutionary dynamic optimization techniques (EDO) comes, to maintain a level of diversity in consecutive populations in a way that doesn't disrupt normal convergence as well as being able to adapt to possible upcoming changes and re-converge to new optimal solution.

Several techniques could be use to make EA able to adapt to environmental changes:

1. Introducing diversity after detecting a change:
   In this approach diversity is increased after detecting a change using hyper-mutation[6], or variable local search [7].
2. Maintaining diversity during the search process:
   This is achieved through fitness sharing[8], thermodynamic GA [9], sentinel placement [10], population-based incremental learning [11], random immigrants [12], and several PSO variants [13] [14][15].
3. Memory based:
   Used when changes are periodical or recurrent, it might be useful to save previously found solutions, which introduced what is called memory in EAs.
4. Prediction methods:
   In some dynamic environments changes may follow some predictable patterns, in which learning models could be used to predict possible changes.
5. Multi-population methods:
   In this method multiple sub-populations are maintained. Each sub-population can handle a separate area of the search space.

Overall, multi-population approach seems to be the most flexible approach which combines the strengths of maintaining diversity, memory and adaptation approaches.

In this paper, we propose a modified HS with multi sub-harmony memories (sub-HMs) and diversity injection to solve the dynamic SP problem in MANETs. In our problem, we are seeking bandwidth-delay-constrained least cost solutions. Both end-to-end delay and link with minimum bandwidth are used as quality-of-service (QOS) metrics to guarantee good performance of real time applications. We conducted a set of experiments to compare between our modified HS, restart HS (RHS), and standard HS (SHS) [16]. The results show that the proposed modified HS out performs both RHS and SHS.

In section 2 we present how the network model representing MANET is generated along with its parameters, in section 3 we describe basic HS algorithm, in section 4 we introduce the proposed modified HS, in section 5 experiments results, and conclusion is presented in section 6.

2. NETWORK MODEL

A MANET is represented by an undirected graph *G (V, U)*, consisting of a set of nodes *V* and a set of links *U* connecting nodes. A link will exist between two nodes if both nodes fall in the transmission range of each other. When first creating the network model, nodes are created and randomly assigned *X* and *Y* position within the workspace dimensions. After calculating the set of neighbors for each node, a random cost within $[C_{min}, C_{max}]$ range and a random bandwidth within $[BW_{min}, BW_{max}]$ range is drawn for each link. After that the source node *s* and destination node *t* are randomly chosen from the set of nodes *V* such that both *s* and *t* are not neighbors, i.e. the minimum path length will be 3 nodes and two

links. Finally, delay upper bound is determined, bandwidth lower bound is determined, the optimization process seeks to get the optimal or near optimal solution which is a bandwidth-delay constrained least cost acyclic path from source to destination.

To simulate topological changes, every time a change is due, some nodes are chosen to sleep and others are chosen to wake up if they are sleeping. Of course, sleeping nodes can't participate in paths. The optimization technique should be able to track new optimal solution every time a change is introduced. Network model parameters are grouped in table 1.

Table 1: Network model parameters

Parameter	Description
$G_0(V_0, U_0)$	Initial network topology
$G_i(V_i, U_i)$	Network topology after the i^{th} change
s	Source node
t	Destination node
$P_i(s, t)$	Path from node s to node t on graph G_i
c_u	Cost of link u
d_u	Propagation delay of link u
bw_u	Bandwidth of link u
$\Delta(P_i)$	Total propagation delay of path P_i
$C(P_i)$	Total cost of path P_i
Δ_p	Delay upper bound
β_p	Bandwidth lower bound

In a formal way, with a MANET $G(V, U)$ and a request to connect source node s to destination node t, with link bandwidth lower bound β_p and delay upper bound Δ_p. It is required that after every topology change to get a series of acyclic paths with the least cost according to eq. 1, delay bounded according to eq.2 and with minimum link bandwidth greater than or equal to bandwidth lower bound according to eq. 3.

$$C(P_i) = min\left\{\sum_{u \in P_i(s,t)} c_u\right\} \quad (1)$$

$$\Delta(P_i) = \sum_{u \in P_i(s,t)} d_u \leq \Delta_p \quad (2)$$

$$\min_{u \in P_i(s,t)} \{bw_u\} \geq \beta_p \quad (3)$$

3. HARMONY SEARCH ALGORITHM

Harmony search (HS) is a population-based meta-heuristic music-inspired optimization algorithm that mimics the improvisation process of musicians searching for a better state of harmony developed by Geem *et al.*[16]. HS is a simple algorithm, it incorporates fewer operations and hence requires less computations. Since its first appearance in 2001 HS has been applied to a lot of optimization problems. Furthermore, experiments and numerical comparisons shown that the convergence of HS was faster than genetic algorithm [17][18][19].

The HS improvisation process consists of the following steps[20]:

1. Initialize HS parameters and harmony memory.
2. Improvise a new harmony vector.
3. Update harmony memory.
4. Check stopping criteria.

The above steps are explained in more details next. A complete HS improvisation process is in figure 1.

3.1. Initialize HS Parameters and Harmony Memory

Harmony memory (**HM**) is a place where solution vectors will be stored during the improvisation process, *HM* is like mating pool of GA.

HS depends on few parameters including harmony memory size (**HMS**) that specifies the number of solutions vectors in the harmony memory, harmony memory consideration rate (**HMCR**) is the probability of considering decision variables values stored in *HM*, pitch adjustment rate (**PAR**) is the probability of randomly modifying selected values from *HM*, and number of improvisations (**NI**) is the number of iteration of *HS*.

$$HM = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_{N-1}^1 & x_N^1 \\ x_1^2 & x_2^2 & \cdots & x_{N-1}^2 & x_N^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^{HMS-1} & x_2^{HMS-1} & \cdots & x_{N-1}^{HMS-1} & x_N^{HMS-1} \\ x_1^{HMS} & x_2^{HMS} & \cdots & x_{N-1}^{HMS} & x_N^{HMS} \end{bmatrix} \quad (4)$$

Algorithm1: HS improvisation process.
Step1: Initialize HS parameters and HM
Step2: Improvise a new harmony x^{new} as follow:

```
1       for( i = 1 : N )
2          if(randNum < HMCR)                    //Memory consideration
3             x_i^new = x_i^k , k ∈(1, 2, …, HMS)
4             if(randNum < PAR)                  // Pitch adjustment
5                x_i^new = x_i^new + randNum*BW  // BW is the amount of change
6             endif
7          else
8             x_i^new = LB_i + randNum*(UB_i - LB_i)  // LB_i and UB_i are lower and upper bounds of variable i
9          endif
10      endfor
```

Step 3: Update HM, $x^{worst} = x^{new}$, if fitness(x^{new}) < fitness(x^{worst})
Step4: If termination criteria reached, stop and return best harmony of HM, otherwise go to step2.

Figure 1: HS improvisation process

3.2. Improvise a New Harmony Vector

Improvising a new harmony vector is the main operation in *HS*. In this operation, the two main parameters *HMCR* and *PAR* are used, so that the new harmony vector is generated both from existing *HM* members as well as randomly selected variables. New harmony $x^{new} = (x_1^{new}, x_2^{new}, \dots, x_N^{new})$, where x_1^{new} is the value of the first decision variable, is generated using 3 rules: (i) memory consideration, (ii) pitch adjustment and (iii) random selection.

Each variable value of the new harmony vector is generated by memory consideration with probability *HMCR* or randomly selected with probability (1 - *HMCR*). Decision variable values generated by memory consideration are subject to pitch adjustment with probability *PAR*.

Having *HMCR* closer to 1.0 is in the favor of choosing from the historic variable values stored in the *HM*, which results in fast convergence. On the other hand, small *HMCR* is in the favor of increasing the diversity of the *HM*. A small value of *PAR* results in passing decision variable values chosen by memory consideration as is which enhances local exploitation around selected variable, while large *PAR* helps with the exploration.

3.3. Update Harmony Memory

After new harmony vector x^{new} is generated, its fitness value is calculated and it replaces the worst harmony in *HM* if its fitness is better than the worst harmony.

3.4. Check Stopping Criteria

If the stopping criteria is met the improvisation process stops and the best harmony vector is retrieved as the optimal or near optimal solution, otherwise the improvisation continues.

4. MODIFIED HS FOR DYNAMIC SP PROBLEM IN MANET

Using population-based meta-heuristic optimization technique to address the SP problem in MANETs is completely different from other continuous optimization problems in many ways, as shown in eq. 4, *HM* is a *HMS* × *N* matrix, where *HMS* is number of solution vectors in *HM* and *N* is the number of decision variables. Solution vectors represent paths from a specific source to a specific destination. These paths are not equal in size, i.e. we may have a path of 2, 3, 4 or more nodes.

Another difference is that when initializing *HM* with harmony vectors representing paths or improvising a new harmony vector, only acyclic paths are accepted. Due to the aforementioned differences and difficulties, some modifications were proposed to *HM* initialization and improvise new harmony vector steps.

Standard HS suffers from difficulties in performing local search, hence some modification were introduced to enhance local search. Pan *et al.* [21] introduced local-best harmony search algorithm

Algorithm2: Modified HS improvisation process.

Step1: Initialize HS parameters and HM
Step2: Initialize sub-HMs
Step3: start modified HS main loop

```
1       for (j = 1 : NI )    //NI is number of improvisations
4         if (j%diversityInjectionRate = = 0)
5           introduceDiversity(diversityAmount)
6         if ( j%regroupRate = = 0)
7           regroupSub_HMs()
8         for (i = 1 : sub-HMsNumber )    //for each sub-HM do
9           if (randNum < HMCR)                    //Memory consideration
10            x^new = (randIndex) , randIndex ∈(1, 2, …, HMS)
11            if (randNum < PAR)                    // Pitch adjustment
12              x^new_randIndex = x^new_randIndex ±BW, randIndex ∈(1, 2, …, N)      // BW is the amount of change
13            endif
14          else
15            x^new= generateNewHarmony ()  //generates a new harmony vector
16          endif
17          replaceWorst (x^new,i)              // if x^new is better than worst harmony in sub-HM no. i
18        endfor
19      endfor
```

Figure 2 Modified HS improvisation process

with dynamic subpopulations (**DLHS**). We adopted their contribution in continuous optimization to our combinatorial optimization SP problem. Pan *et al.* divided the *HM* into small sized *sub-HMs*, each *sub-HM* evolve independently. To prevent pre-mature convergence to local optima, the *sub-HMs* are regrouped every specified number of iteration to exchange the search experience with each other.

The process of improvising new harmony vector is changed to minimize the possibility of producing infeasible paths, also a diversity injection mechanism is introduced to balance topological changes and to help algorithm search process. Steps of our modified HS are summarized in figure 2 and details are presented next.

4.1. Initialize *HM*

HM is initialized with solution vectors representing paths from source to destination. To create a path we start with source node then choose randomly a node from source node neighbors then the neighbor is considered in turn and a node of its neighbors is randomly selected, such that it does not already exist in path so far, until the destination node is reached. For this process to work, graph representing network model must be connected, so we repeat creating network model until a connected graph is reached.

4.2. Divide *HM* Into *Sub-HMs*

After initialization, HM is divided randomly into equal *sub-HMs*, as stated above *sub-HMs* are regrouped every specified number of iteration to exchange search experience.

4.3. Improvise a New Harmony Vector

For every *Sub-HM* a new harmony vector is improvised and it replaces the worst harmony in sub-*HM*. Following the traditional process would lead to a lot of cyclic paths. Also, it is somehow difficult to produce a new harmony vector since paths are not equal in length. Our new harmony vector improvisation process as shown in Figure 2 lines 9-16, is to select at random a harmony vector from *HM* with probability *HMCR* and change one of its nodes randomly with probability *PAR*, or generate a completely new harmony from outside the *HM* which represents random selection. Using the proposed process, there is still a possibility that a cyclic path is produced. Two common procedures are known to deal with infeasible solutions, either to add a penalty to the infeasible path fitness value which helps the HS algorithm to get rid of this infeasible path or to repair the path. Both ways were tested and we found out that path repair is time consuming and may not work all the time, as sometimes during the improvisation process some nodes may not have neighbors due to some

topological changes which leads to a path that can't be repaired or if repaired it will a completely different path other than the path before repair, on the other hand adding penalty guarantees that HS will get rid of the infeasible solutions at some point in the improvisation process. That way we used a fitness function that adds penalty for infeasible paths.

4.4. Evaluation of Solutions

We adopted R. Forsati *et al.* [20] fitness function shown in eq. 5, because it adds a penalty for any path with total delay greater than delay upper bound, and/or with minimum link bandwidth less than bandwidth lower bound.

$$F\big(P_i(s,t)\big) = \frac{\alpha}{C(P_i)} \times \varphi\big(\Delta(P_i) - \Delta_p\big) \times \Psi\big(\min linkBW(P_i) - \beta_p\big) \quad (5)$$

where α is a positive real factor, $\varphi(x)$ and $\Psi(x)$are penalty functions as follow:

$$\varphi(x) = \begin{cases} r_1 . r_D , & x > 0 \\ 1 , & x \le 0 \end{cases} \quad (6)$$

$$\Psi(x) = \begin{cases} r_2 . r_B , & x < 0 \\ 1 , & x \ge 0 \end{cases} \quad (7)$$

It is clear that $\varphi(x)$ and $\Psi(x)$ are equal to 1 when the path satisfies delay and bandwidth constraints; if any path doesn't satisfy one or both constraints, $\varphi(x)$ and $\Psi(x)$ are equal to $r_1 . r_D$ and $r_2 . r_B$, respectively, where $0 < r_1, r_2, r_B, r_D < 1$. $r_B$and r_D are exact constants for adjusting importance of constraints, where $r_1$and r_2 are self-adaptation penalty factors defined in eq. 8 and 9,

$$r_1 = r_{1\,min} + \frac{n}{NI}\,(r_{1\,max} - r_{1\,min}) \quad (8)$$

$$r_2 = r_{2\,min} + \frac{n}{NI}\,(r_{2\,max} - r_{2\,min}) \quad (9)$$

where *NI* is number of improvisations and n is current number of iterations, and $r_{1\,min}$, $r_{1\,max}$, $r_{2\,min}$, $r_{2\,max} \in [0,1]$.

4.5. Diversity Injection

To balance the effect of topological change and to enhance the ability of HS to find new optimal solutions, a level of diversity should be maintained. Diversity level of the whole *HM* is maintained through creating new harmony vectors using only random selection. Diversity is injected in a rate that doesn't disrupt normal convergence as well as help the HS find new optimal solutions. After creating the new harmony vector, a random index is generated within [1, *HMS*] range, if that random index is not that of best harmony vector in *HM* the new harmony vector replaces the harmony with that random index, otherwise the new harmony vector replaces the worst harmony vector in *HM*.

5. EXPERIMENT RESULTS AND ANALYSIS

We conducted 7 experiments to measure and compare our proposed modified HS to solve dynamic SP problem in MANETs. In all our experiments, to simulate dynamic environments some nodes are scheduled to sleep every change interval, i.e. if the change rate is 10 and the change severity is 2, then every 10 iteration 2 nodes will be set to sleep, that means they won't be used to create paths and the paths they were part of will become infeasible. To make the change more sever, whenever a change is due we choose to set to sleep nodes on the current best path. Source node and destination node can't be scheduled in any of these changes. Also previously sleeping nodes are wakened right before the due change.

For all our experiments, we created a network model of 100 nodes in a 100 * 100 meters workspace. Each node with a transmission range 20 meters. Choosing small transmission range is in the favor of testing the real nature of MANETs in which nodes act as routers to rely data between nodes out of each other transmission range. We set the link cost range to [1, 10], bandwidth range to [1, 10] Mbps, delay upper bound equals to 3 times the diagonal propagation delay, and the bandwidth lower bound is set to 2 Mbps.

For the modified HS common parameters, we set the *HMS* to 100, *sub-HMS* to 10 i.e. we will have 10

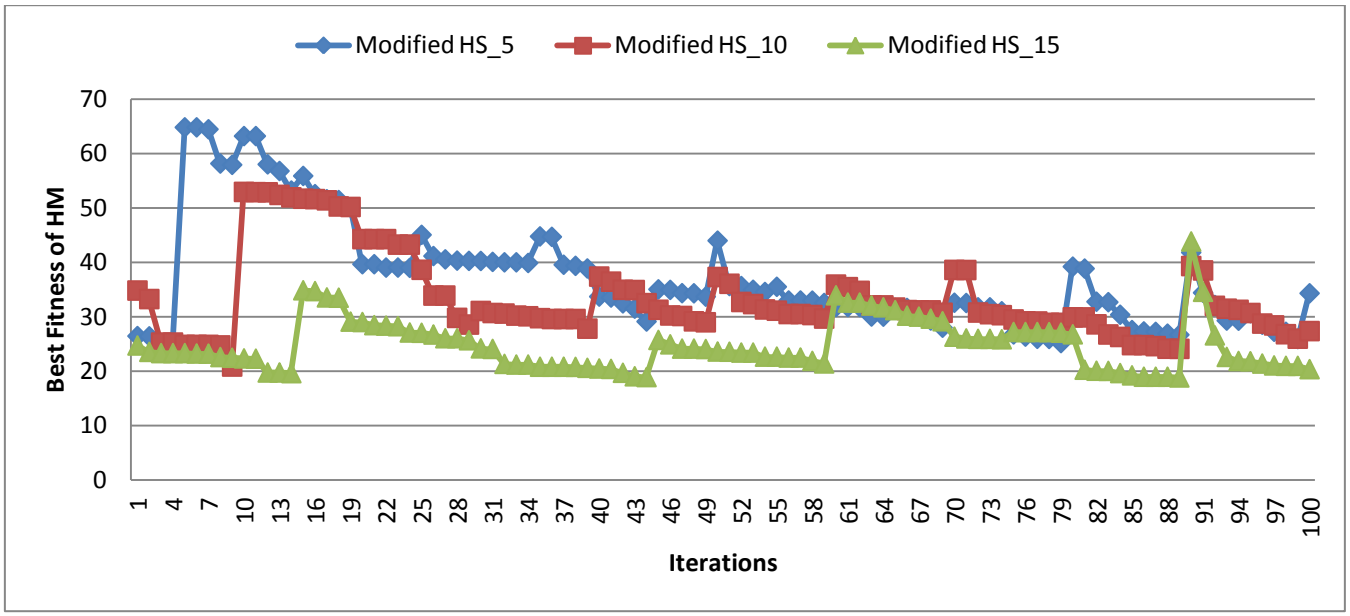

Figure 3: Compare solution quality of Modified HS with different change rates: 5, 10, and 15

sub-HMs, *NI* to 100, and *sub-HM* regroup rate to 10 iterations. The remaining parameters change rate, change severity and diversity injection rate, are used to test the impact of their different values on the algorithm.

We ran each experiment 20 times, and then we got the average of these runs.

We conducted 3 experiment sets:

5.1. Impact of The Change Rates

Using the above parameters, with the diversity injection rate set to 15 and diversity amount set to 1, and experimenting with change rates of 5, 10, and 15 the results in figure 3 shows that high change rates don't give a room for HS to converge to good solutions before the next due change, while increasing the change rate to 10 then to 15 i.e. slower changes, a noticeable improvement to the solution

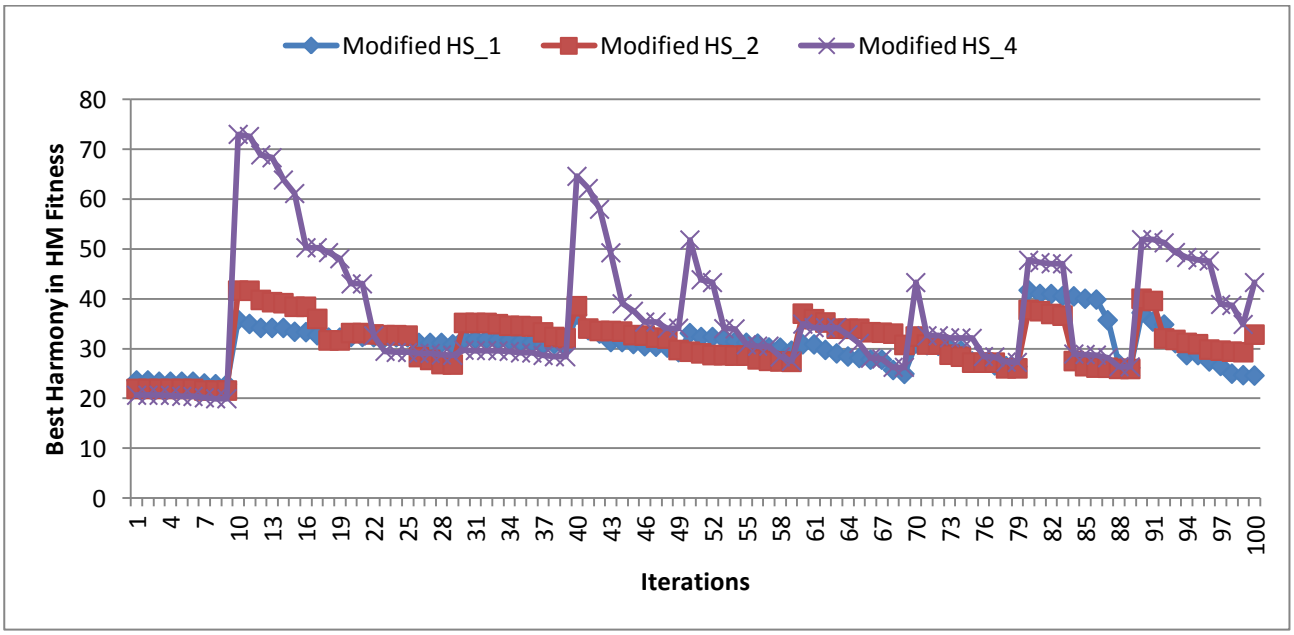

Figure 4: Compare solution quality of Modified HS with different best path change severity: 1, 2, 4 nodes

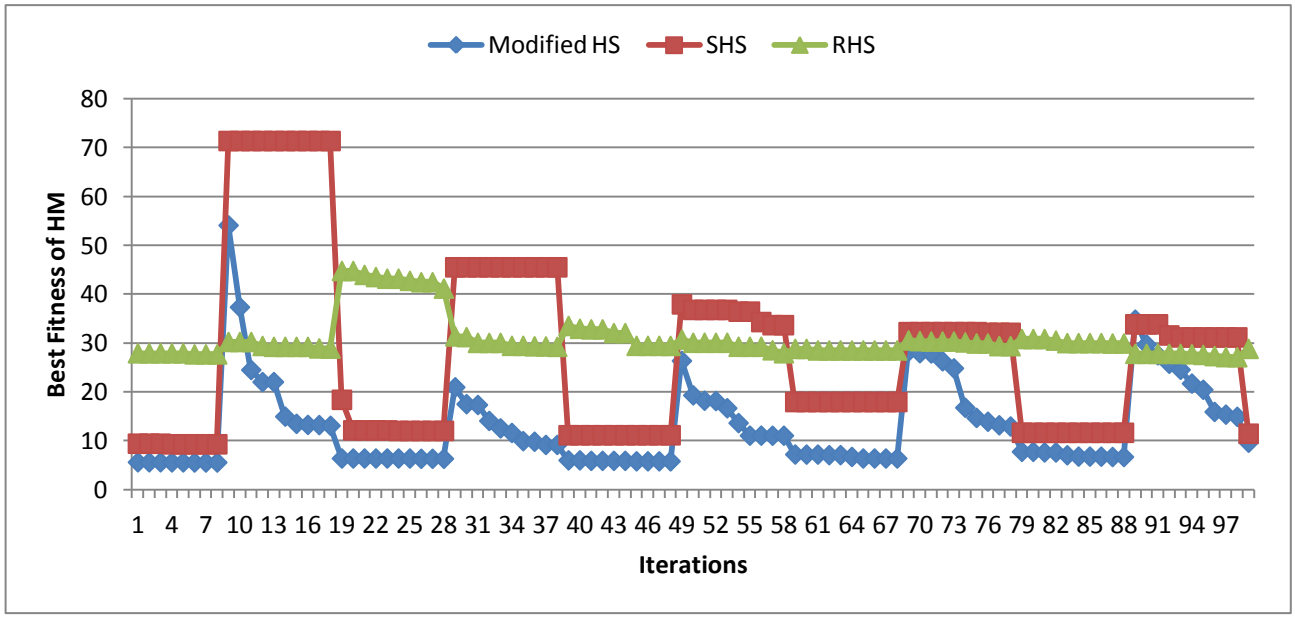

Figure 5: Compare solution quality of modified HS, SHS, and RHS, where change severity is 4, change rate is 10

quality is achieved since the HS has more time to converge to good solution before next change.

5.2. Impact of Change Severity

With the same parameters in section 5.1, but the change rate is fixed to 10 iterations. We tested the impact of the change severity with values 1, 2, and 4 nodes on best path. As figure 4 shows, as the change severity increases the ability of the algorithm to re-converge to a good solution before the next change is decreasing. On the other hand and as expected change severity 4 possesses the worst behavior. This happened because setting 4 nodes to sleep made the current best path infeasible as well as some of nearby good solutions became infeasible as well which represents a big damage that requires more time for the algorithm to recover.

5.3. Compare Between Modified HS, Standard HS (SHS), and Restart HS (RHS)

The third experiments set is to compare between modified HS, SHS, and RHS in terms of convergence speed and solution quality. Having changes happening every 10 iterations, a level of diversity should be introduced to help algorithm find new optimal solutions. We test the impact if after change: improvisation process just continues which is represented by SHS, the whole HS is restarted represented by RHS, and modified HS that introduces diversity amount of 1every 5 iterations. Change severity of modified HS is 4 nodes, SHS and RHS use *HMCR* = 0.9, and *PAR* = 0.2.

As shown in figure 5 our modified HS outperforms SHS in terms of fast re-convergence and good quality of solutions. Modified HS possesses the best solution quality and faster convergence after changes. We can notice that SHS and modified HS are giving the same behavior but modified HS converges faster to better quality solutions due the amount of diversity in its *HM*. The best paths in RHS are always within a fitness range that seems to be fixed and don't converge this happens because of the continuous restart i.e. the whole *HM* is recreated every change interval which gives no chance for the HS algorithm to converge to better solutions before next change also because the previously attained search experience is wasted every time the HS is restarted.

6. CONCLUSION

We studied and analyzed the dynamic SP problem in MANETs, and proposed a modified HS algorithm to solve this dynamic optimization problem. Solving SP problem in a dynamic environment is more

challenging than in stationary environments. So our proposed algorithm follows the most promising way of maintaining diversity in a HM which is using sub-HMs with diversity injection mechanism that balances the effect of frequently introduced change. Our experiments showed that the proposed algorithm out performs SHS, and RHS. Our future work will focus on conducting more experiments on the modified HS and apply it on more DOPs.

7. REFERENCES

[1] Subir Kumar Sarkar, T. G. Basavaraju, and C. Puttamadappa, *Ad Hoc Mobile Wireless Networks: Principles, Protocols and Applications*. Boston, MA, USA: Auerbach Publications, 2007.

[2] Hui Cheng and Shengxiang Yang, "Genetic algorithms with elitism-based immigrants for dynamic shortest path problem in mobile ad hoc networks," in *Proc. IEEE Congress Evolutionary Computation CEC '09*, 2009, pp. 3135-3140.

[3] L. Barolli, A. Koyama, and N. Shiratori, "A QoS routing method for ad-hoc networks based on genetic algorithm," in *Proc. 14th Int Database and Expert Systems Applications Workshop*, 2003, pp. 175-179.

[4] Vinay Kolar, Nael B. Abu-Ghazaleh, and Petri Mahonen, "Decomposition for low-complexity near-optimal routing in multi-hop wireless networks," in *ICC'09: Proceedings of the 2009 IEEE international conference on Communications*, Piscataway, NJ, USA, 2009, pp. 5302-5307.

[5] Trung Thanh Nguyen, Shengxiang Yang, and Juergen Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, no. 0, pp. 1-24, 2012. [Online]. http://www.sciencedirect.com/science/article/pii/S2210650212000363

[6] Helen G Cobb, "An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments," *Technical Report AIC-90-001*, 1990.

[7] Frank Vavak, KA Jukes, Terrence C Fogarty, and others, "Performance of a genetic algorithm with variable local search range relative to frequency of the environmental changes," *Genetic Programming*, pp. 22-25, 1998.

[8] Andersen H. C., "An investigation into genetic algorithms and the relationship between speciation and the tracing of optima in dynamic functions," Queenslan University of Technology, Honours thesis November 1991.

[9] Naoki Mori, Hajime Kita, and Yoshikazu Nishikawa, "Adaptation to a changing environment by means of the thermodynamical genetic algorithm," in *Parallel Problem Solving from Nature—PPSN IV.*: Springer, 1996, pp. 513-522. [Online]. http://link.springer.com/chapter/10.1007/3-540-61723-X_1015

[10] Ronald W Morrison, *Designing evolutionary algorithms for dynamic environments.*: Springer, 2004.

[11] Shengxiang Yang and Xin Yao, "Experimental study on population-based incremental learning algorithms for dynamic optimization problems," *Soft Comput.*, vol. 9, no. 11, pp. 815-834, 2005.

[12] John Grefenstette, "Genetic Algorithms for Changing Environments," in *Parallel Problem Solving from Nature 2*, 1992, pp. 137-144.

[13] S. Janson and M. Middendorf, "A hierarchical particle swarm optimizer and its adaptive variant," *IEEE_J_SMCB*, vol. 35, no. 6, pp. 1272-1282, 2005. [Online]. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1542271

[14] Tim M Blackwell, Peter J Bentley, and others, "Dynamic search with charged swarms," in *Proceedings of the genetic and evolutionary computation conference*, 2002, pp. 19-26.

[15] Tim and Branke, Jurgen Blackwell, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE_J_EVC*, vol. 10, no. 4, pp. 459-472, 2006. [Online]. http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1665033

[16] Zong Woo Geem, *Music-Inspired Harmony Search Algorithm, Theory and Applications*, Zong Woo Geem, Ed.: Springer-Verlag Berlin Heidelberg, 2009.

[17] M Mahdavi, M Fesanghary, and E Damangir, "An improved harmony search algorithm for solving optimization problems," *Applied mathematics and computation*, vol. 188, no. 2, pp. 1567-1579, 2007. [Online]. http://www.sciencedirect.com/science/article/pii/S0096300306015098

[18] Kang Seok Lee and Zong Woo Geem, "A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice," *Computer methods in applied mechanics and engineering*, vol. 194, no. 36, pp. 3902-3933, 2005. [Online]. http://www.sciencedirect.com/science/article/pii/S0045782504004682

[19] Kang Seok Lee, Zong Woo Geem, Sang-ho Lee, and Kyu-woong Bae, "The harmony search heuristic algorithm for discrete structural optimization," *Engineering Optimization*, vol. 37, no. 7, pp. 663-684, 2005. [Online]. http://www.tandfonline.com/doi/abs/10.1080/03052150500211895

[20] R Forsati, AT Haghighat, and M Mahdavi, "Harmony search based algorithms for bandwidth-delay-constrained least-cost multicast routing," *Computer Communications*,

vol. 31, no. 10, pp. 2505-2519, 2008. [Online]. http://www.sciencedirect.com/science/article/pii/S0140366408001941

[21] Q.K. Pan, PN Suganthan, JJ Liang, and M.F. Tasgetiren, "A local-best harmony search algorithm with dynamic subpopulations," *Engineering Optimization*, vol. 42, no. 2, pp. 101-117, 2010. [Online]. http://www.tandfonline.com/doi/abs/10.1080/03052150903104366