Ad Hoc Networks 58 (2017) 198–204

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

A binary artificial bee colony algorithm for constructing spanning trees in vehicular ad hoc networks

Xin Zhang[a,b], Xiu Zhang[a,b,*]

[a] College of Electronic and Communication Engineering, Tianjin Normal University, Tianjin, China
[b] Tianjin Key Laboratory of Wireless Mobile Communications and Power Transmission, Tianjin Normal University, Tianjin, China

ARTICLE INFO

Article history:
Received 3 May 2016
Revised 21 June 2016
Accepted 3 July 2016
Available online 5 July 2016

Keywords:
Artificial bee colony
Vehicular ad hoc network
Minimum spanning tree
Binary representation

ABSTRACT

To accomplish reliable and efficient information routing, strong paths connecting all nodes are required in vehicular ad hoc networks (VANETs). Classical algorithms in graphic theory could find only one minimum spanning tree (MST) in VANETs. Swarm intelligence paradigms are able to obtain several alternatives to MST, which is useful for improving reliability of VANETs. This paper proposes a binary coded artificial bee colony (BABC) algorithm for tackling the spanning tree construction problem. A two-element variation technique is designed to keep the consistence of binary coded solutions. The proposed algorithm is applied to tackle a roadside-to-vehicle communication example. The success rate and average hitting time of the algorithm to find MST are also analyzed. It is found that the BABC algorithm could find MST with 92% probability. Though it is slower than Kruskal algorithm in terms of computational time, the BABC algorithm can attain several suboptimal spanning trees in one run. This suggests that the algorithm would be useful under the condition that tree paths are required to be rebuilt frequently while the network topology is unchanged in a short period.

1. Introduction

There are two types of wireless networks — base-station oriented networks and ad hoc wireless networks. Hybrid wireless network is formed by these two networks [1]. The ad hoc network topology is desirable because of its low cost, plug-and-play convenience, and flexibility. Its usage of bandwidth and battery power is more efficient. Due to the lack of infrastructure, the data is forwarded to the destination via a multi-hop fashion. Generally, the ad hoc network has been studied in optimization [2], target detection [3,4], etc. In some scenarios, a set of base stations are connected by wired links and placed within the ad hoc networks to form a wired infrastructure, intending to enhance the whole network performance. This resulting network is referred to as a hybrid wireless network. Due to the dynamic nature of such network, quite often computational intelligence approaches such as fuzzy logic systems [5,6], swarm intelligence could be applied to optimization in hybrid wireless networks. This paper aims to apply artificial bee colony (a recently created swarm intelligence paradigm) to vehicular ad hoc network (VANET).

Along with the development of intelligent transportation systems (ITSs), VANET becomes a major network communication topic in ITSs. VANET supports communication among arbitrary vehicles, which is realized by wireless communication and data transmission techniques. The network organizes vehicles and roadside transport facilities in order to let all customers receive and send real time traffic-related information. Moreover, VANET can enhance traffic efficiency and also assure road safety and traveling comfort. VANET is an important branch of mobile ad hoc network (MANET) with specific properties. In vehicular ad hoc networks (VANETs), vehicles are considered as nodes. VANETs include roadside-to-vehicle communication (RVC), inter-vehicle communication (IVC), and hybrid vehicular communication (HVC). Roadside unit equipment is usually deployed at service station, dangerous road intersection or slippery roads [7]. Also a global positioning system device is often placed in a node of VANETs to provide location information.

Routing plays an important role in VANETs. A good routing scheme presents reliable performance in receiving and sending message from a source node to a destination node [8]. Clearly, to accomplish reliable message routing, a sufficient number of nodes is required to constitute strong paths from source nodes to destination nodes. Existing researches mainly concentrate on the routing quality in VANETs to extend network lifetime. However, little effort has been spent to efficiently constructing routing trees. Generally, the efficiency of a distributed algorithm is assessed by runtime and the quantity of exchanged messages among the nodes [9]. Due to the failure of sensor nodes, fast moving vehicles, or battery

* Corresponding author.
E-mail addresses: tjnumark@126.com (X. Zhang), zhang210@126.com (X. Zhang).

http://dx.doi.org/10.1016/j.adhoc.2016.07.001
1570-8705/

exhaustion, VANETs require a routing tree being constructed as fast as possible, and runtime becomes an important measure.

As above mentioned, efficient minimum spanning tree (MST) construction is very useful in routing problem of VANETs. Moreover, clustering is a helpful technique in improving the network scalability. MST could be applied to find the ideal path from cluster head to base station in wireless sensor networks. Classical MSP algorithms in graphic theory include Kruskal algorithm, prim algorithm, sollon algorithm and so on. All of them are greedy methods, and they could generate one and only MST. In VANETs, some nodes may escape the scope of base station or roadside unit, and their reliability is in low level. In this case, finding a group of MST or nearly minimum spanning trees is more applicable than classical algorithms. Swarm intelligence algorithms are inspired from collective behaviors of natural swarms. They are good at searching a group solutions for optimization problems with no prior knowledge.

Artificial bee colony (ABC) is one of the commonly used swarm intelligence paradigms. In the beginning, it is designed to handle continuous problems [10]. To efficiently deal with MST problem, this paper designs a binary coded ABC algorithm. A novel search equation is proposed to avoid producing out of bound solutions. Moreover, the average hitting time of the proposed algorithm for solving MST problem is discussed in this paper.

The rest of the paper is organized as follows. Section 2 reports three types of VANET architecture and spanning tree problem. Section 3 presents the proposed algorithm and its analysis. Section 4 reports the simulation results compared with Kruskal algorithm. The average hitting time to optimal solution of spanning tree problem is also discussed in this section. Section 5 gives the conclusion.

2. VANETs architecture and minimum spanning tree problem

Ad hoc network is a new networking mode but is difficult to analyze. Gao et al. analyzed the local delay of ad hoc networks with large and small primary traffic [11]. Gorrieri et al. analyzed the performance of clustered vehicular sensor networks, and found some insights in the trade-off between decision delay and energy efficiency [12]. Cornejo et al. studied a reliable neighbor discovery layer for MANETs, which could be applied in regular square region or hexagonal tiling region [13]. Han et al. investigated the relay placement and power allocation for cooperative relaying [14]. Lachowski et al. implemented distributed algorithms for building routing spanning trees in wireless sensor networks [15]. Kponyo et al. studied a VANET cluster-on-demand MST based on prim algorithm [16]. Wang and Liang investigated node transmission limit of hybrid wireless networks, a base station is in charge of long distance communications of nodes [17].

The architecture of VANETs can be classified to three prototypes: RVC, IVC, and HVC. In RVC, vehicles communicate with server through roadside access point or base station to acquire information. Due to the fast moving property of vehicles in VANETs, each node frequently escapes from current access point and breaks into the covering field of another access point, which is usually called handover problem. Suppose a node lies in covering fields of more than one access points, how to choose a proper access point to reach maximum transmission gain is another problem to be solved in RVC.

In IVC, a vehicle could acquire necessary information from other vehicles when access points or base stations are unavailable. Communication routing is the main problem in this type. IVC routing includes unicast routing, cluster-based routing, geocast routing, and broadcast routing.

HVC is a more realistic scene in daily life. If a vehicle runs out the service scope of roadside access points and base stations, it could connect to server through other vehicles, which are considered as routers. This is an economic solution than deploying access points or base stations.

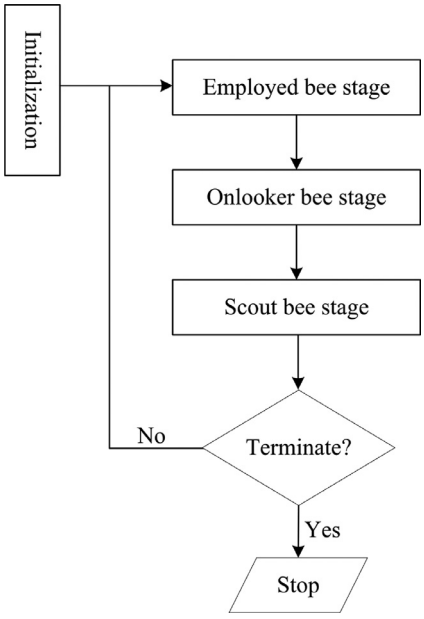


Fig. 1. Flow chart of standard ABC algorithm.

The MST problem has been studied for decades, researchers still attempt to design an algorithm to tackle it in linear time [18]. The problem is described as follows. Given an undirected graph $G = (N, E, W)$, where $W = \sum_{e \in E} w_e$ is weight function of defined on edge set E. A tree $T = (N, E_T, W_T)$ is called a spanning tree if it contains all nodes in graph G, where $W_T = \sum_{e \in E_T} w_e$ is the weight function of T. Among all spanning acyclic trees of G, the one having the least total weight is called minimum spanning tree of G. For convenience, G is assumed to be a connected tree in this paper.

In real world applications, different MST represents different design schemes. A spanning tree having nearly minimal total weight to MST may also be valuable to practitioners based on specific conditions. Classical MSP algorithms could not provide several candidate spanning trees. Also it is impractical to search candidate spanning trees by brute force method. Swarm intelligence approaches show good performance in dealing with combinatorial optimization problems. They can obtain a set of (nearly) optimal solutions in a short time. ABC is taken in this paper to tackle the MST problem in VANETs.

3. Binary artificial bee colony

The colony of standard ABC consists of three kinds of honey bees. They are employed bees, onlooker bees, and scout bees, hence the algorithm is comprised of three stages as shown in Fig. 1. They work one after another to constitute a cycle in ABC as illustrated in Algorithm 1. n_e is the number of employed bees. *limit* is the number of consecutive iterations that a food source fails to be improved. Generally, the number of food sources is n_e, so is the number of onlooker bees. The number of scout bees depends on *limit* and the algorithm's search status. In standard ABC algorithm, onlooker bees conduct local exploitation to refine solutions, scout bees perform global exploration to detect promising search area, while the search by employed bees is in the middle. It can either be exploration if food sources were scattered widely in search space, or exploitation if food sources were close to each other.

Algorithm 1: Procedures of the ABC algorithm.

Input: $\boldsymbol{f}(\cdot)$, D, $\mathbf{x}^{min}$, $\mathbf{x}^{max}$, n_e, $limit$, $feval = 0$
Output: a set of optimal solutions obtained by the algorithm

```
randomly initialize n_e food sources, feval = feval + n_e;
evaluate the function values of the initialized food sources, and compute their fitness values;
while termination criteria are not satisfied do                                   // main cycle
    for i ← 1 to n_e do                                                           // employed bee phase
        employed bee i searches around food source i based on some rules and produces v_i;
        evaluate v_i, feval = feval + 1;
        greedy selection between v_i and x_i;
    end
    for i ← 1 to n_e do                                                           // onlooker bee phase
        onlooker bee i chooses a food source based on the fitness of food sources;
        onlooker bee i searches around the chosen source k based on some rules and produces v_i;
        evaluate v_i, feval = feval + 1;
        greedy selection between v_i and x_k;
    end
    for i ← 1 to n_e do                                                           // scout bee phase
        if food source i has not been improved in the last limit iterations then
            a scout bee flies out and randomly explores the search space to produce x_i';
            replace food source i by x_i'
        end
    end
end
```

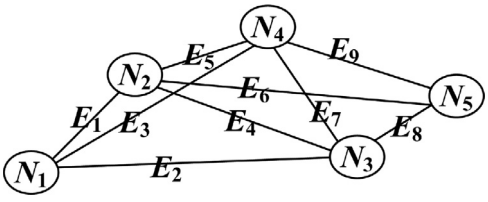


Fig. 2. Example of initializing a binary string.

Standard ABC is targeted for continuous optimization problems. To deal with the MST problem, binary representation is utilized. Each element corresponds to an edge in graph G and takes value 0 or 1, where value 0 means the associated edge is not included in spanning tree T, and value 1 means the opposite. Each edge $e \in E$ is assigned an integer flag so that all edges are coded into a binary string of length $|E|$. Thus, a tree T can be built given such kind of binary string.

A simple heuristics is applied to quickly produce a spanning tree. Because a spanning tree T connects all nodes without loop, it must contain $|N| - 1$ edges. Thus, a binary bit string stands for a spanning tree under the condition that it contains $|N| - 1$ 1s in the string. This heuristics is used in the initialization step as well as search steps in main cycle.

In initialization, a bit string of length $|E|$ is created with each bit set to 0. Then randomly choose $|N| - 1$ positions and set those to 1. For example, given a graph $G = (N, E)$ with $|N| = 5$ and $|E| = 9$ as shown in Fig. 2, where the nodes and edges are numbered in order so that each bit of a solution could be decoded to an edge. Fig. 3 presents the procedure of generating an initial solution. The initialized string could be decoded to a tree having $|N| - 1$ edges, which is a spanning tree if all nodes are connected, or a tree with loop. Accordingly, the associated tree is shown in Fig. 4, which is a spanning tree of graph G.

Following the above heuristics, a two-element variation technique is proposed for the search of employed bees and onlooker bees. First, a position is randomly chosen and denoted as i_1. Second, another position i_2 is also randomly chosen depending on the value of i_1. If $i_1 = 0$, then i_2 is selected from the elements with 1s; otherwise, i_2 is selected from those with 0s. This technique is shown in Fig. 5. In this figure, graph G contains 5 nodes and 9 edges, hence bit string length $L = 9$. It is observed that the number of elements with value 0 or 1 does not change, hence this technique always produces a candidate string with 5 elements with value 1.

If the *limit* flag of some food sources is satisfied, the associated food sources would be discarded, and are replaced by new ones randomly exploration by scout bees. The procedure is similar to the initialization step. Elitism is NOT used in scout bee phase. That is the best solution could be abandoned by honey bees, but the best solution is memorized at the end of each cycle.

The pseudo code of binary ABC (BABC) algorithm is similar to standard ABC as in Algorithm 1 except food sources are encoded by bit strings and $D = L$.

To deal with the hybrid flowshop problem, a discrete ABC algorithm is designed combining differential evolution and variable neighborhood search methods [19]. Pan et al. also presented a discrete ABC algorithm for hybrid flowship problem, which used a combination of forward decoding and backward decoding methods [20]. A bitwise operation (e.g., XOR operator '$\wedge$', or operator '|' and operator '&') based binary ABC was proposed for continuous function optimization [21]. ABC was also modified to optimize binary halftone patterns of images [22]. Ozturk et al. proposed to use similarity information of solutions to improve the performance of binary ABC and verified it on dynamic clustering [23]. Instead of designing binary ABC, Kiran applied continuous ABC to handling binary optimization problems [24].

Similar to existing works, binary representation is used in BABC. The variation search shown in Fig. 5 is different from previous researches. Because BABC is designed to handle spanning tree problems, the number of 1 elements during variation search has to be constant. Moreover, the generation of initial solutions is also specific to spanning tree problems.

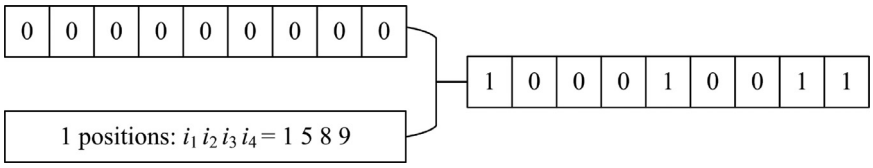


Fig. 3. Example of initializing a binary string.

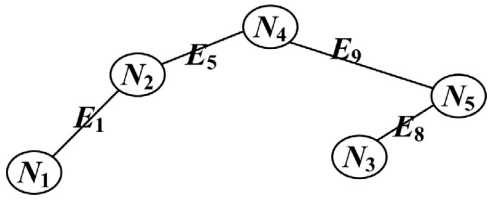


Fig. 4. Tree structure represented by a binary string as in Fig. 3.

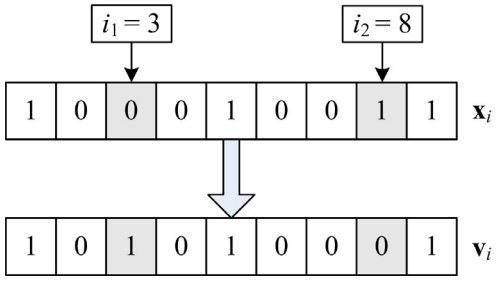


Fig. 5. Example of two-element variation technique.

4. Simulation

In this section, the proposed BABC algorithm will be investigated by an MST example of RVC type networks.

4.1. Test example and Simulation setting

A graph G of 16 nodes are taken as an example in VANETs as shown in Fig. 6. This example includes 16 nodes and 32 edges. Among the 16 nodes, one is a base station and the others are cars in a traffic way. The number of edges $|E|$ is. The weight function is Euclidean distance between two nodes. The objective function of a binary string is the sum of weights over all edges. The MST problem is turned into a minimization model.

The parameter setting of the BABC algorithm is:

(1) $n_e = 10$;
(2) $limit = 100$.

The algorithm is independently executed 25 times to obtain its average performance. The soundness of this parameter setting has already been demonstrated as in [25,26]. It terminates when either the following conditions is met:

(1) the maximum number of function evaluations (MFE) is reached, where $MFE = 10000$;
(2) $|f(\mathbf{x}) - f(\mathbf{x}^*)| \leq 10^{-8}$, where $f(\mathbf{x})$ stands for the best value found by BABC and $f(\mathbf{x})^*$ is the optimal value found by Kruskal algorithm.

Average hitting time means the average running time of BABC reaching optimal value. The following metric is used, which is also called expected running time (ERT) [27]:

$$E(RT(\Delta f)) = \hat{E}(FE^s) + \frac{1-\hat{p}^s}{\hat{p}^s}\hat{E}(FE^u), \quad (1)$$

Table 1
Summary of spanning trees found by the BABC algorithm in 25 runs.

ST	Weight	Edge series	Runs
1	1949.25	$E_1\ E_4\ E_7\ E_8\ E_9\ E_{10}\ E_{14}\ E_{17}\ E_{18}\ E_{20}\ E_{25}\ E_{26}\ E_{27}\ E_{28}\ E_{32}$	23
2	1949.35	$E_1\ E_4\ E_7\ E_8\ E_9\ E_{10}\ E_{14}\ E_{17}\ E_{18}\ E_{20}\ E_{25}\ E_{26}\ E_{28}\ E_{30}\ E_{32}$	2

Table 2
A set of spanning trees found by the BABC algorithm in a typical run.

ST	Weight	Edge series
1	1949.25	$E_1\ E_4\ E_7\ E_8\ E_9\ E_{10}\ E_{14}\ E_{17}\ E_{18}\ E_{20}\ E_{25}\ E_{26}\ E_{27}\ E_{28}\ E_{32}$
2	1950.11	$E_1\ E_4\ E_6\ E_7\ E_8\ E_{10}\ E_{14}\ E_{17}\ E_{18}\ E_{20}\ E_{25}\ E_{26}\ E_{27}\ E_{28}\ E_{32}$
3	2035.04	$E_1\ E_4\ E_5\ E_7\ E_{10}\ E_{12}\ E_{14}\ E_{18}\ E_{20}\ E_{22}\ E_{25}\ E_{26}\ E_{27}\ E_{28}\ E_{32}$
4	2078.48	$E_1\ E_3\ E_5\ E_8\ E_9\ E_{10}\ E_{14}\ E_{17}\ E_{18}\ E_{20}\ E_{25}\ E_{26}\ E_{27}\ E_{28}\ E_{32}$
5	1958.72	$E_1\ E_4\ E_5\ E_7\ E_{10}\ E_{12}\ E_{14}\ E_{17}\ E_{18}\ E_{20}\ E_{25}\ E_{26}\ E_{27}\ E_{28}\ E_{32}$
6	2154.08	$E_1\ E_2\ E_5\ E_7\ E_9\ E_{10}\ E_{15}\ E_{17}\ E_{18}\ E_{20}\ E_{24}\ E_{26}\ E_{28}\ E_{31}\ E_{32}$
7	2554.57	$E_2\ E_3\ E_5\ E_7\ E_{10}\ E_{11}\ E_{12}\ E_{18}\ E_{20}\ E_{22}\ E_{24}\ E_{26}\ E_{27}\ E_{30}\ E_{31}$
8	2423.31	$E_1\ E_3\ E_6\ E_7\ E_8\ E_{11}\ E_{15}\ E_{17}\ E_{18}\ E_{20}\ E_{23}\ E_{25}\ E_{27}\ E_{30}\ E_{31}$
9	2297.14	$E_1\ E_3\ E_4\ E_5\ E_9\ E_{10}\ E_{14}\ E_{17}\ E_{21}\ E_{22}\ E_{25}\ E_{26}\ E_{27}\ E_{30}\ E_{32}$
10	2220.11	$E_2\ E_4\ E_6\ E_7\ E_8\ E_{10}\ E_{13}\ E_{17}\ E_{18}\ E_{20}\ E_{24}\ E_{26}\ E_{27}\ E_{28}\ E_{32}$

where $RT(\Delta f)$ denotes the runtime of the algorithm reaching $\Delta f = 10E^{-8}$; $\hat{E}(FE^s)$ means the average number of functions evaluations that those runs successfully reach Δf; $\hat{E}(FE^u)$ is the average number of function evaluations in the unsuccessful runs; $\hat{p}^s$ is the ratio of runs that reaching Δf over 25 runs. In this paper, $\hat{E}(FE^u) = 10,000$.

The simulation is conducted on a PC with 4-core 2.50GHz CPU and 4GB of memory. Thus, a fair environment can be made for runtime comparison.

4.2. Simulation results

First, Kruskal algorithm is applied to handle the example and the resulting MST is shown in Fig. 7. Since this method always finds MST of an undirected acyclic graph, the solution is considered as $\mathbf{x}^*$. Its objective function value $f(\mathbf{x}^*) = 1949.25$. The computational time of Kruskal algorithm is about 0.05 s.

Table 1 presents a summary of the results attained by the BABC algorithm over 25 runs. In the table, BABC successfully finds the same MST solution as Kruskal algorithm in 23 out of 25 runs, which is 92%. Although in the remaining two runs, the algorithm fails to reach the same solution, the difference is negligible in terms of the weights of spanning trees. The spanning tree represented by the second solution in Table 1 is plotted in Fig. 8. It can be seen that node N_{12} serves as a relay node for N_{15} and N_{16} in Fig. 7, while the relay node changes to N_{14} in Fig. 8. In case that N_{12} is unstable and becomes unavailable, without performing another spanning tree search, the solution as in Fig. 8 could take over the information communication if N_{14} keeps stable.

The average number of function evaluations (FE) is 3219 in the 23 successful runs. Clearly, the success rate of BABC reaching MST is 92%. The average hitting time computed by (1) is about 4088 in 25 runs, which means that the BABC algorithm could find the optimal solution costing 4088 FEs on average. The corresponding computer running time is about 0.32 second.

Table 2 presents the results attained by the BABC algorithm in a typical run. It is observed that BABC can obtain ten candidate

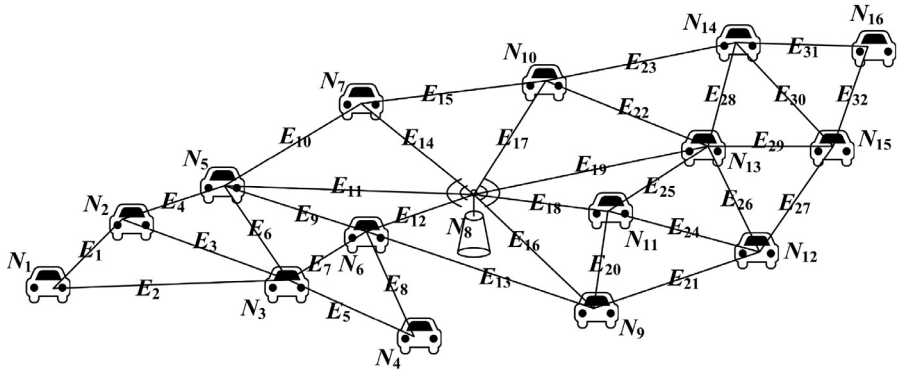


Fig. 6. A VANET example with 16 nodes and 32 edges.

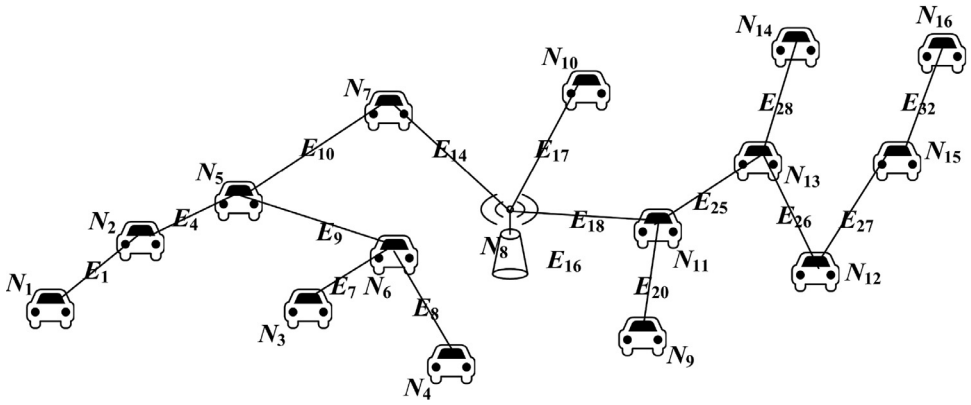


Fig. 7. Minimum spanning tree found by the Kruskal algorithm.

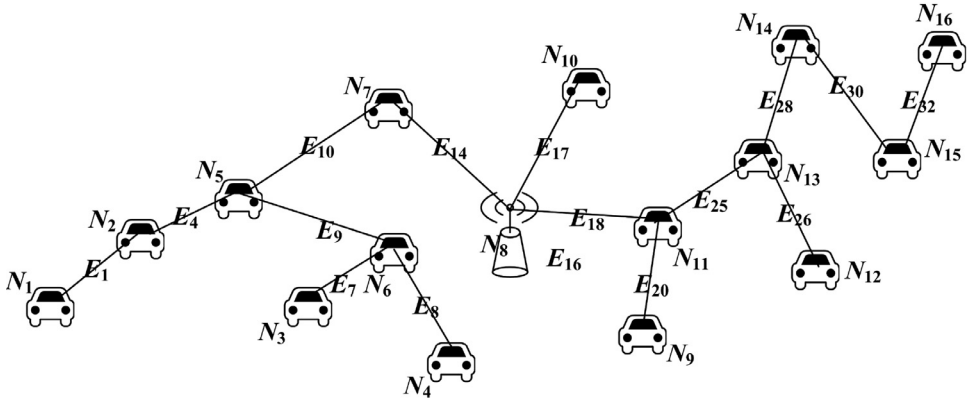


Fig. 8. The second minimum spanning tree found by the BABC algorithm.

spanning trees for users in one run, due to $n_e = 10$. See from the table, several good spanning trees could be attained by executing BABC once. This is useful under the condition that spanning tree needs rebuilding frequently yet the network topology is unchanged. BABC consumes more runtime than Kruskal algorithm, though, it might be a good choice considering reconstructing MST repeatedly.

The BABC algorithm is also apt to deal with optimal routing from a source node to a destiny node, which is equivalent to shortest path problem. Take the base station N_8 as starting node, while N_1 or N_{16} is taken as destiny node, respectively. For verifying the performance of the proposed algorithm, the optimal path from N_8 to N_1 or N_{16} are manually determined as shown in the third column of Table 3. ERT of the algorithm can then be computed. The results are given in Table 3.

Table 3
Shortest path test of the BABC algorithm in 25 runs.

Source	Destiny	Shortest path	Runs	ERT
N_8	N_1	N_8 N_5 N_2 N_1	25	251.80
N_8	N_{16}	N_8 N_{13} N_{14} N_{16}	25	409.24

It can be seen from Table 3 that the success rates of BABC finding optimal path from N_8 to N_1 or to N_{16} are 100%. The ERT for path N_8 to N_1 is about 251.8 over 25 runs, which means that the BABC algorithm could find shortest path costing about 252 FEs on average. The ERT for path N_8 to N_{16} is about 409.24 over 25 runs, which means that the algorithm could find shortest path costing about 410 FEs on average. Although the proposed algorithm can be applied to deal with optimal routing in VANETs, it has a large variance based on the ERT values.

5. Conclusion

Efficient and reliable communications in VANETs requires constructing strong paths among vehicles. Due to the failure of sensor nodes, fast moving vehicles, or battery exhaustion, some nodes may become unavailable of ad hoc network. New paths has to be built to assure information communication in ad hoc network. Path construction can be modeled as minimum spanning tree (MST) problem in graphic theory. Classical algorithms in graphic theory could find MST but cannot provide alternatives for usage. Note that the artificial bee colony (ABC) paradigm is able to accomplish this target.

This paper proposes a ABC algorithm with binary representation (BABC). VANETs are represented by undirected acyclic graphs. Edges of a graph are encoded to a binary string in BABC. To efficiently exploring all possible spanning tree combinations, a two-element variation technique is designed to keep the number of edges constant in a binary string.

Tested on an example of roadside-to-vehicle communication networks, the BABC algorithm finds the optimal spanning tree as using Kruskal algorithm. Moreover, BABC could also produce candidate suboptimal spanning trees. These trees are useful for path construction when some nodes in ad hoc network becomes unavailable. Although the BABC algorithm costs more computational time than Kruskal algorithm, it would be a useful tool when routing paths have to be build repeated in a short time. Furthermore, the algorithm can also be applied to deal with shortest path problem in VANETs.

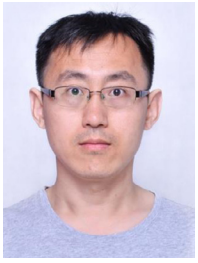
The average hitting time of the BABC algorithm could be reduced by designing more effective search equations or by identifying search patterns in binary strings. For networks with a large number of nodes, it may be better to add prior knowledge about network structure to enhance the efficiency of BABC. These issues will be further studied in the future.

Acknowledgements

This research was supported in part by the Applied Basic Research Program of Tianjin (15JCYBJC51500, 15JCYBJC52300), and the Tianjin Thousand Youth Talents Plan Project of Tianjin Normal University (ZX110023).

References

[1] X. Wang, Q. Liang, On the throughput capacity and performance analysis of hybrid wireless networks over fading channels, IEEE Trans. Wireless Commun. 12 (6) (2013) 2930–2940.
[2] Q. Liang, X. Cheng, S. Huang, D. Chen, Opportunistic sensing in wireless sensor networks: theory and applications, IEEE Trans. Comput. 63 (8) (2014) 2002–2010.
[3] Q. Liang, Radar sensor wireless channel modeling in foliage environment: UWB versus narrowband, IEEE Sensors J. 11 (6) (2011) 1448–1457.
[4] Q. Liang, X. Cheng, S. Samn, NEW: Network-enabled electronic warfare for target recognition, IEEE Trans. Aerosp. Electron. Syst. 46 (2) (2010) 558–568.
[5] Q. Liang, Situation understanding based on heterogeneous sensor networks and human-inspired favor weak fuzzy logic system, IEEE Syst. J. 5 (2) (2011) 156–163.
[6] Q. Liang, X. Cheng, KUPS: Knowledge-based ubiquitous and persistent sensor networks for threat assessment, IEEE Trans. Aerosp. Electron. Syst. 44 (3) (2008) 1060–1069.
[7] S. Bitam, A. Mellouk, S. Zeadally, HyBR: A hybrid bio-inspired bee swarm routing protocol for safety applications in vehicular ad hoc networks (VANETs), J. Syst. Arch. 59 (10) (2013) 953–967.
[8] R. Lachowski, M.E. Pellenz, M.C. Penna, E. Jamhour, R.D. Souza, An efficient distributed algorithm for constructing spanning trees in wireless sensor networks, Sensors 15 (1) (2015) 1518–1536.
[9] M. Khan, G. Pandurangan, V.S.A. Kumar, Distributed algorithms for constructing approximate minimum spanning trees in wireless sensor networks, IEEE Trans. Parallel Distrib. Syst. 20 (2009) 124–139.
[10] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithms, J. Global Optim. 39 (3) (2007) 459–471.
[11] J. Gao, C. Yin, Analysis of the local delay with slotted-aloha based cognitive radio ad hoc networks, EURASIP J. Wireless Commun. Netw. 2015 (1) (2015) 1–6.
[12] A. Gorrieri, M. Martalo, S. Busanelli, G. Ferrari, Clustering and sensing with decentralized detection in vehicular ad hoc networks, Ad Hoc Netw. 36, Part 2 (2016) 450–464.
[13] A. Cornejo, S. Viqar, J.L. Welch, Reliable neighbor discovery for mobile ad hoc networks, Ad Hoc Netw. 12 (2014) 259–277.
[14] L. Han, J. Mu, W. Wang, B. Zhang, Optimization of relay placement and power allocation for decode-and-forward cooperative relaying over correlated shadowed fading channels, EURASIP J. Wireless Commun. Netw. 2014 (1) (2014) 1–7.
[15] R. Lachowski, M.E. Pellenz, M.C. Penna, E. Jamhour, R.D. Souza, An efficient distributed algorithm for constructing spanning trees in wireless sensor networks., Sensors 15 (1) (2015) 1518–1536.
[16] J.J. Kponyo, Y. Kuang, E. Zhang, K. Domenic, VANET cluster-on-demand minimum spanning tree (MST) prim clustering algorithm, in: International Conference on Computational Problem-solving (ICCP), 2013, 2013, pp. 101–104. Jiuzhai, China
[17] X. Wang, Q. Liang, On the throughput capacity and performance analysis of hybrid wireless networks over fading channels, IEEE Trans. Wireless Commun. 12 (6) (2013) 2930–2940.
[18] S. Pettie, V. Ramachandran, An optimal minimum spanning tree algorithm, J. ACM 49 (1) (2002) 16–34.
[19] Z. Cui, X. Gu, An improved discrete artificial bee colony algorithm to minimize the make span on hybrid flow shop problems, Neurocomputing 148 (2015) 248–259.
[20] Q.-K. Pan, L. Wang, J.-Q. Li, J.-H. Duan, A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation, Omega 45 (2014) 42–56.
[21] D. Jia, X. Duan, M.K. Khan, Binary artificial bee colony optimization using bitwise operation, Comput. Ind. Eng. 76 (2014) 360–365.
[22] A. Chatterjee, B. Tudu, K.C. Paul, Binary grayscale halftone pattern generation using binary artificial bee colony (bABC), Sig. Image Video Process. 7 (6) (2013) 1195–1209.
[23] C. Ozturk, E. Hancer, D. Karaboga, Dynamic clustering with improved binary artificial bee colony algorithm, Appl. Soft Comput. 28 (2015) 69–80.
[24] M.S. Kiran, The continuous artificial bee colony algorithm for binary optimization, Appl. Soft Comput. 33 (2015) 15–23.
[25] M.A. Contreras-Cruz, V. Ayala-Ramirez, U.H. Hernandez-Belmonte, Mobile robot path planning using artificial bee colony and evolutionary programming, Appl. Soft Comput. 30 (2015) 319–328.
[26] X. Zhang, X. Zhang, S.L. Ho, W.N. Fu, A modification of artificial bee colony algorithm applied to loudspeaker design problem, IEEE Trans. Magnetics 50 (2014) 737–740.
[27] N. Hansen, A. Auger, R. Ros, S. Finck, P. Pošík, Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009, in: Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation, in: GECCO '10, ACM, New York, NY, USA, 2010, pp. 1689–1696.



Xin Zhang received the B. Sc., M. Sc., and Ph.D. degrees from Ludong University, Shandong University of Science and Technology, City University of Hong Kong, in 2006, 2009, and 2013, respectively. He is currently a Lecturer in the College of Electronic and Communication Engineering, Tianjin Normal University.
His main research interests are swarm intelligence, communication network optimization, evolutionary computation, and machine learning. He has published more than 15 technical papers on these subjects, including more than eight international journal papers.

Xiu Zhang received the B. Eng. and M. Eng. degrees in biomedical engineering from the Hebei University of Technology, Tianjin, China, in 2006 and 2009, respectively. Her research during master period concerns the analysis of the electroencephalograph signal when the magnetic signal is used to stimulate the acupoints in human subjects. She received the Ph.D. degree and completed the postdoctoral research work in electrical engineering from The Hong Kong Polytechnic University in 2012 and 2015, respectively.
She is now a lecturer in Tianjin Normal University. She has published about 20 papers in the IEEE TRANSACTIONS. Her research interests mainly focus on numerical methods of electromagnetic field computation, novel wireless energy transfer systems, and wireless network optimization.