2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Evolutionary Path Planning of a Data Mule in Wireless Sensor Network by Using Shortcuts

Shao-You Wu and Jing-Sin Liu

Abstract—**Data collection problem of generating a path for a data mule (single or multiple mobile robots) to collect data from wireless sensor network (WSN) is usually a NP-hard problem. Thus, we formulate it as a Traveling Salesman Problem with Neighborhoods (TSPN) to obtain the possibly short path. TSPN is composed of determinations of the order of visiting sites and their precise locations. By taking advantage of the overlap of neighborhoods, we proposed a clustering–based genetic algorithm (CBGA) with an innovative way for initial population generation, called Balanced Standard Deviation Algorithm (BSDA). Then, effective shortcut schemes named Look-Ahead Locating Algorithm (LLA) and Advanced-LLA are applied on the TSPN route. By LLA, a smoother route is generated and the data mule can move while ignoring about 39% clusters. Extensive simulations are performed to evaluate the TSPN route in some aspects like LLA hits, LLA improvement, Rotation Degree of Data Mule (RDDM), Max Step and Ruggedness.**

Keywords: Traveling salesman problem with neighborhood, Path planning, Clustering, Genetic algorithm, Shortcut, Data collection.

I. Introduction

Traveling salesman problem (TSP) is first defined by Irish and British mathematicians in 1800s. It describes that a salesman wants to travel many cities for promoting products and decides the proper (usually, the shortest) path to visit all cites, finally back to the original city. TSP is known as a NP-hard problem. Hence, some optimization approaches are proposed: Particle Swarm Optimization Algorithms (PSO), Ant Colony Optimization (ACO), Simulated Annealing (SA), Artificial Immune Algorithm (AI), Tabu Search (TS) and Genetic Algorithm (GA). In this paper we adopted GA, for its sensitivity of the environmental parameters and the ease to be implemented. TSP with neighborhood (TSPN) is a variant of TSP [1], [2]. In TSPN, the visiting city no longer counts as a point but an area, referred as neighborhood. That is, the ordering of visiting and the entry point to a city is considered simultaneously. In summary, two subproblems are involved in TSPN: (1) determination of the visiting order of sites and (2) determination of precise point of sites. Recently, there are increasing attentions to the study of TSPN motivated from data collection problem in wireless sensor network (WSN) using data mules [3]-[8]. WSN consists of numerous distributed autonomous sensors which can monitor environmental conditions [9]. Sensors can actively forward collecting data to the sink, called multi-hop forwarding approach. Or, supervisor can employ data mules to passively execute the data gathering, called one-hop forwarding approach. Basically, the former has single failure problems leading to overload at some sensors. So, our research targeted at the latter and tried to use some mechanisms to shorten the path of the data mule. In one-hop forwarding approach planned path to traverse WSN indeed significantly reduces energy consumption and buffer overflow compared with commonly-used multi-hop forwarding approach, but it pays for the increased data latency.

Pairwise-disjointness of neighborhoods is often assumed for some earlier contributions of theoretical interests, e.g. constant factor approximation algorithms. In recent works, taking the overlap of neighborhoods of two nodes into account, constant factor approximation algorithms [5] and (meta)heuristic algorithms are developed for the path planning to minimize latency. We attributed the data gathering task in WSN to a TSPN with sensing range being accessible area (neighborhood). To reduce the path length, we proposed a two-phase method named "clustering-based genetic algorithm (CBGA)" for path planning of data mule [10] resulting good solutions of TSPN. The first phase is to divide the network into small disjoint clusters by using a clustering algorithm (CA), which is flexible and efficient. The second phase is to decide the visiting order of all clusters and the precise entry points of clusters by using genetic algorithm (GA) [11], [12]. The minor implementation variations of GA could yield different achievable performance. In our recent research [10], extensive simulations show SCX (sequential constructive crossover) [13] in combination of 2-opt [15] in GA achieves the best average performance. This paper presents a further improvement of data gathering route. One is that we design an effective method, named Balanced Standard Deviation Algorithm (BSDA), to generate initial population in GA. Besides, CBGA is combined with a non-evolutionary shortcut heuristics, called Look-Ahead Locating Algorithm (LLA), which is based on the combine-skip-substitute (CSS) scheme [3]. Moreover, Advanced-LLA

S. Y. Wu and J. S. Liu are with Institute of Information Science, Academia Sinica, Nangang, Taipei, Taiwan 115, ROC. (e-mail: bluechip993@gmail.com, liu@iis.sinica.edu.tw).

(ALLA) is designed for some situations unfit for LLA. This shortcut heuristics is feasible to optimize the path length, derive a smoother path and reduce the transmission delay and orienting errors of data mule.

The paper is organized as follows. The model and problem statement are described in section II. Section III introduces the clustering-based genetic algorithm. An innovative method of initial population generation, called Balanced Standard Deviation Algorithm (BSDA), is contained in Section IV. Section V presents the shortcut scheme, Look-Ahead Locating Algorithm (LLA) and its advanced version (Advanced-LLA). Section VI shows the simulation results and analysis. Finally, the conclusion is made in section VII.

II. Model and Problem Statement

A. Model

A data mule and a set of stationary sensors $\boldsymbol{S} = \{s_1, s_2, s_3 \ldots s_n\}$ constitute the wireless sensor network (WSN). All devices in WSN are also equipped with omnidirectional antennae for locating. Sensors are randomly deployed over a two-dimensional plane and their precise locations are known in advance. Each sensor s_i can transmit signals within a disk region (sensing radius). When data mule is in the sensing radius of a sensor, it can receive data from that sensor. In this paper, we assume the data mule has unlimited power and memory to collect all data in WSN.

B. Problem Statement

To minimize the route length of data mule, the data-gathering problem is formulated as a special case of Traveling Salesman Problem with Neighborhoods (TSPN) where the neighborhood is the common intersection area of each cluster. TSPN includes the visiting of all neighborhoods and the visiting order of clusters. Because TSPN is a NP-hard problem, we adopt Genetic Algorithm (GA) [11] to find the optimal. Furthermore, shortcut schemes, denoted as Look-Ahead Locating Algorithm (LLA), and its advanced version (Advanced-LLA, ALLA), are employed. Finally, the data mule follows the generated path and starts to collect data of all clusters from the start point (sink), returning back to the sink while collecting completes.

III. Clustering-Based Genetic Algorithm

This paper is based on one-hop data-gathering scheme and hence, all sensors must independently communicate with data mule. As shown in Fig. 1, the clustering-based genetic algorithm (CBGA) is proposed for generating a good TSPN route. First, Clustering Algorithm (CA) is executed to efficiently group possibly most sensors into disjoint clusters; that is, a cluster may contain some sensors and a sensor will not belong to different clusters. CA also gives each cluster a waypoint located among the common intersection area and then data mule can communicate with each sensor belonging to that cluster at this waypoint. CA permitted data mule to collect data by visiting all waypoints instead of every sensor. Meanwhile, Closest Position Algorithm (CP algorithm), which is a waypoint selection scheme and can further reduce path length, is also given. In core genetic algorithm, various ways were suggested to generate initial population such as Chromosome Generation Algorithm (CGA), Random Initial (RI) and an innovative way, called Balanced Standard Deviation Algorithm (BSDA). About crossover and mutation operators of GA, several distinct operations are tested and analyzed by our previous study. The experimental results showed that SCX (or MSCX) accompanied by 2-opt has the best performance. Therefore, SCX and 2-opt in genetic algorithm were used in this paper for further research.

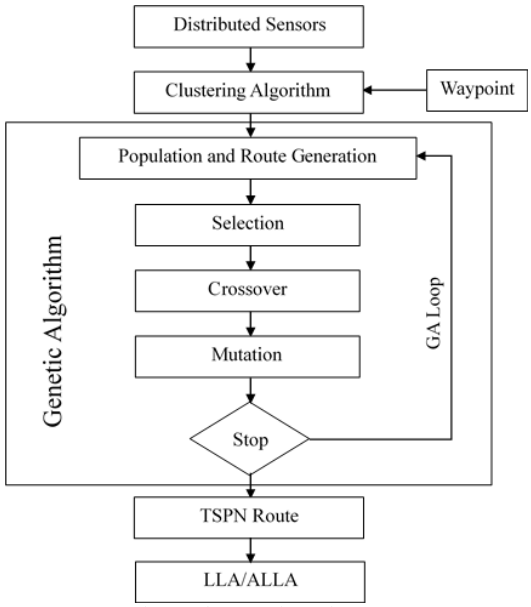
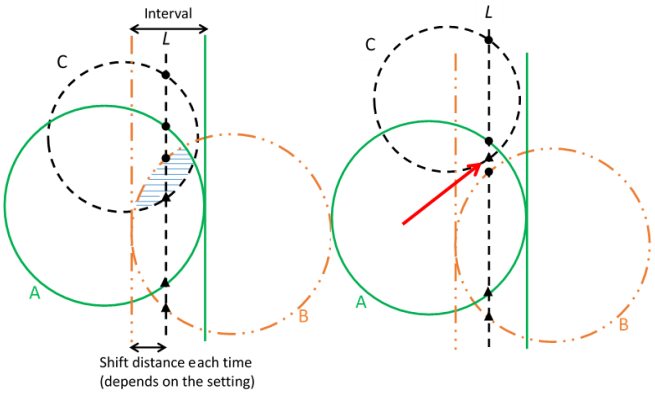


Fig. 1. The overview of CBGA.

A. Clustering Algorithm

In order to lower the route length, CA groups some sensors into a cluster and in this way, data mule just needs to visit the clusters and collect multiple data from diverse sensors at a single fixed waypoint in the common intersection area of a cluster.



(a) (b)

Fig. 2. Upper and lower crossing points are denotes as circular and triangulation points, respectively. (a) Uppers are higher than lowers and hence, form a cluster whose common intersection area is marked as diagonal lines. (b) A lower point is higher than uppers and cannot form a cluster.

CA works by checking whether there is a common intersection area among some sensors, and whether the sensors form a cluster; then, assigning a waypoint and cluster ID to the cluster. In Fig. 2 (a), three sensors (A, B, C) with different sensing radius and two lines (left-dotted line and right-solid line) are set while A and B are grouped into a cluster. Now, CA checks whether C is allowed to merge with this cluster. First, left line shifts toward the right side (denoted as L) but not exceeds the right line. Then, count crossing points of A, B and C; each sensor at most has two crossing points (upper/lower point). In Fig. 2 (a), all upper points are higher (in y axis) than lower ones that indicates the existence of common intersection area (marked as diagonal lines). Hence, it means the three sensors can constitute a cluster with waypoint being:

(x, y)=(x_axis[L], y_axis[(MIN(uppers)+MAX(lowers))/2]).

The moving-checking operation ends while left line exceeds right line or until the conditions of forming a cluster meet. Another case shown in Fig. 2 (b), there is a lower point higher than upper point and hence, these three sensors do not share a common intersection area and cannot form a cluster. CA is designed to create disjoint clusters because if a sensor is assigned to more than one cluster at the same time, data mule will collect duplicate data at different waypoint.

B. Find the Closest Position (CP) for Each Cluster

CP algorithm is a waypoint selection scheme [10] and is embedded in crossover, mutation and fitness calculating operations. While there are some candidate entry points, CP will choose the closest one. Experimental results show CP saves about 6% path length no matter how the density of WSN is.

C. Chromosome Encoding and Generation of Initial Population

Any order containing all clusters without repetition is regarded as a valid chromosome and each gene is encoded by cluster ID. In order to generate the initial population without repeated chromosomes, three methods are used in this paper: Chromosome Generation Algorithm (CGA), Random Initial (RI) and Balanced Standard Deviation Algorithm (BSDA). Fig. 3 shows how CGA works. CGA chooses some locations of the chromosome initially and permutes the clusters of these locations. N chosen clusters cause N! distinct permutations at most. Finally, put the clusters in permuted order back to original locations and new chromosomes are generated. RI is based on an input chromosome; for each gene, RI randomly selects another gene to be swapped with. Moreover, BSDA is designed for generating a robust population with diverse chromosomes because more variability of chromosomes promotes the effect of crossover operations. It is believed that high diversity facilitates the evolution of GA. BSDA will be introduced in details in section IV.

D. Fitness

Each chromosome stands for a possible path and is evaluated by the fitness value. We define the fitness value of i-th chromosome as $F_i = 1/L_i$, where L_i is total length of the visiting path of data mule. A higher fitness value indicates better chromosome; that is, the visiting order is of high quality.

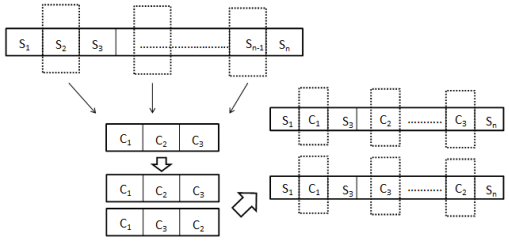


Fig. 3. The demonstration of CGA.

E. Selection

We select two parents to produce one offspring in each crossover operation. First, two separate parents are randomly selected from the population; and the one with higher fitness value was assigned as parent 1. Parent 2 is decided in the same way. The two parents must be different. Then, Parent 1 and parent 2 will be the underlying chromosomes of crossover operation.

F. Crossover

Crossover operation is a probabilistic process and is important in GA because the offspring is produced by how parent chromosomes interact with each other. It is supposed to preserve good segments of parents and even add high quality segments for offspring. We have tested some crossover methods which can mainly be classified as two aspects, purposeless and purposeful. Purposeless crossover operations such as CX (cycle crossover) [16], CSEX (the complete sub-tour exchange crossover) [17] and N-point crossover only provide a trick for generating different permutation of visiting order. Oppositely, SCX (sequential constructive crossover) and MSCX (modified sequential constructive crossover) [13], [14] are types of purposefulness; the inheritance and selection of segments for offspring followed some principles like segment length. SCX was proposed in 2010 by Zakir H. Ahmed as a crossover method in genetic algorithm. Then, MSCX altered the mechanism of selecting segments but remained the advantages and computational efficiency of SCX. We found that purposeful methods with length-oriented principles are better options to our demands. SCX and MSCX could preserve appropriate segments of parents and explore new ones not existing in parents for child chromosomes. In this paper, SCX is adopted.

The following is an example:
In Fig. 4, there are three chromosomes (parent 1, parent 2 and child) with their visiting tracks. Each gene (S, A-H) stands for a cluster data mule has to visit with S as the starting cluster. Table 1 shows the distance matrix of nine clusters (S, A-H), and the values are straight-line length between two different waypoints. According to Table 1, total path length of parent 1 and parent 2 is 34.56 and 39.89, respectively.
SCX starts from cluster S, and "current cluster" is set as S at first. Child chromosome is empty in the beginning. To generate child chromosome, each parent from its visiting

order selects another cluster not appearing in child chromosome currently. Next, two selected clusters are coupled with S to form two segments. Then, the length of two segments is compared. In this example, segment{SA} in parent 1 are smaller than segment{SE} in parent 2; hence, segment{SA} is added to child chromosome. Now, current cluster is A. In the next iteration, segment{AB} is selected. In case of no subsequent cluster from the current one like cluster D in parent 2 (next cluster, S is chosen before). SCX follows the order of cluster ID (from A to H) to find a cluster not appearing before. In this example, segment{DC} is picked out and is shorter than segment{DE} in parent 1. SCX repeats above operation until a new chromosome (child) is produced. After SCX, child chromosome usually has shorter path length (30.15) than its parents. Moreover, it contains segment{DC} and segment{FS}, which are not involved in parents. To some degree, SCX boosts the diversity by discovering new and better segments.

Table 1. Distance matrix of nine waypoints (S, A-H).

	S	A	B	C	D	E	F	G
A	2.94	------	------	------	------	------	------	------
B	2.61	1.55	------	------	------	------	------	------
C	5.78	3.74	3.18	------	------	------	------	------
D	5.33	4.3	3.07	1.89	------	------	------	------
E	5.79	5.72	4.24	3.9	2.02	------	------	------
F	3.11	5.09	3.79	6.06	4.68	3.97	------	------
G	8.55	8.77	7.28	6.56	4.83	3.05	6.04	------
H	9.09	8.01	6.89	4.56	3.82	3.77	7.73	3.88

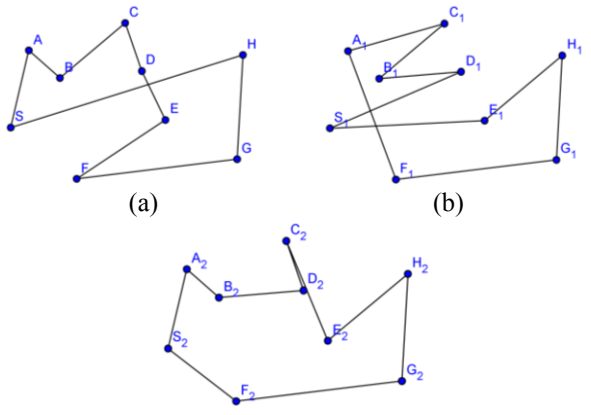


Fig. 4. Two parents chosen by selection operation and a child chromosome generated by SCX. (a) Parent 1: [s a b c d e f g h] = 34.56 (b) Parent 2: [**s** e h g f a c b d] = 39.89 (c) Child: [**s** a b d c e h g f] = 30.15.

G. Mutation

The mutation operation is also a probabilistic process in GA for triggering diversity of population. Mutation operation can extend the search space and avoid getting stuck in local optimum. 2-opt was used for mutation operation. 2-opt is a simple local search algorithm first proposed by Croes in 1958 [15] for solving the TSP. The main idea behind it is to randomly select two segments in a chromosome and swap the two segments to form a new one. If the fitness value of new chromosome is higher, replace current chromosome with it. As shown in Fig. 5, the visiting order of route between b and d is reversed after the 2-opt move. Because of the characteristics of randomness, 2-opt is helpful for GA converging towards a global optimum.

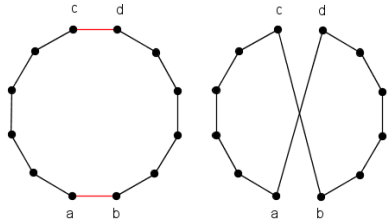


Fig. 5. 2-opt move. (a) original tour is [-cd-ba-]. Two segments cd, ba are selected (b) resulting tour [-cb-da-] based on exchange of two segments by swapping cd with cb and swapping ba with da.

IV. Balanced Standard Deviation Algorithm

Balanced Standard Deviation Algorithm (BSDA) is used to generate initial population of GA. The main idea of BSDA is to lower the standard deviation (SD) of all different segments and promote the diversity of population. Lower SD means that the number of every segment is almost the same. Modulus (%) operator is adopted to derive all possible gene sequences in a balanced way. BSDA needs a prime number (**N1**) to be the base of modulus. Another number **M**, set to 1 initially and ranges from 1 to N1-1, serves as an increment. If the number of clusters (**N2**) is a prime number, N1 is set to N2; otherwise, N1 will be the prime number less than and closest to N2. Assume a population is composed of **P** chromosomes and single chromosome is defined as [C_0, C_1, …, C_{N2-2}, C_{N2-1}]. N2 means there are N2x(N2-1) different segments at most and single chromosome possesses N2 segments. To calculate the SD value of a certain population, we map all appearing segments into a N2xN2 matrix of which fields F_{ii}, i=0~N2-1 are useless. Add 1 to a certain field while the corresponding segment appears. Hence, total value of the matrix will be N2xP; average of that is denoted as SD_avg. We define SD value of a population as following:

$$SD = \sqrt{Total/(N2 * (N2 - 1)) - (SD\_avg * SD\_avg)}. \quad (1)$$

$$Total = \sum(F_{ij} * F_{ij}).$$

where F_{ij} is the value of field (i,j), i, j=0~N2-1, i≠j. (2)

$$SD\_avg = (N2 * P)/(N2 * (N2 - 1)) = P/(N2 - 1). \quad (3)$$

Stage 1:

C_0=0, C_i=((C_{i-1})+M)%N1, i=1~N1-1. By modulus operator, genes which start from C_{N1} will lead to the repetition of previous ones. In this way, invalid chromosomes are generated. Hence, we set C_i=i, i=N1~N2-1. So, every chromosome will possess similar sequences (C_{N1}~C_{N2-1}) which increases SD and lowers the diversity.

Stage 2:
To avoid that, BSDA swaps each of them with another gene which randomly selected from the whole chromosome. After a chromosome is generated, M changes into M%(N1-1)+1 and above operations are repeated until P chromosomes are generated.
We explain how it works by an example as following:
If N2 is 9, not a prime number, the closest prime number, N1, is set to 7. A population with 100 chromosomes is needed. The generated chromosomes are like:

1st: [0, 1, 2, 3, 4, 5, 6, **7**, **8**] => [0, 1, 2, 3, **8**, **7**, 6, 5, 4], M=1
2nd: [0, 2, 4, 6, 1, 3, 5, **7**, **8**] => [0, 2, 4, **7**, 1, 3, 5, **8**, 6], M=2
3rd: [0, 3, 6, 2, 5, 1, 4, **7**, **8**] => [0, 3, 6, 2, **8**, 1, **7**, 4, 5], M=3
4th: [0, 4, 2, 3, 4, 5, 6, **7**, **8**] => [0, 4, 2, **7**, 4, 5, 6, **8**, 3], M=4
5th: [0, 5, 3, 1, 6, 4, 2, **7**, **8**] => [0, 5, **7**, 1, 6, 4, **8**, 3, 2], M=5
6th: [0, 6, 5, 4, 3, 2, 1, **7**, **8**] => [0, 6, 5, 4, 3, **7**, 1, **8**, 2], M=6
7th: [0, 1, 2, 3, 4, 5, 6, **7**, **8**]
……
100th: [0, 4, 2, 3, 4, 5, 6, **7**, **8**]

Modulus operation makes M loop in a fixed range. In this example, 1st and 7th will be the same at first stage, similarly, 2nd and 8th, 3rd and 9th, etc. The best case of BSDA occurs while N1=N2 and P is an integral multiple of N1-1; in that moment, SD is 0. When SD is 0, the population is most balanced and diversiform. The experiment designed to verify the effectiveness of BSDA was described in section VI.

V. Look-Ahead Locating Algorithm and Advanced-Look-Ahead Locating

In this section, based on the CSS (combine-skip-substitute) scheme [3], a short-cut procedure called Look-Ahead Locating Algorithm (LLA) was devised to refine the generated route of GA so that the path length can be shortened further. According to the experimental results, LLA is very powerful not just in shortening path but also in decreasing the total rotated degree of data mule and data communication delay while data gathering is conducting. Fig. 6 shows the overview of LLA. There are two clusters (cluster 1 and 2) and data mule needs to collect all data of these two clusters and goes to C. Data mule has moved from previous location to current location. Now, just consider how it moves from current location to B. By the original method, data mule will follow the black solid lines guided by CP algorithm. That is, it first visits A and turns to B, and two segments are needed. On the contrary, LLA can reach B without visiting middle location A. First, LLA will skip cluster 1 to look ahead cluster 2 and establish a straight line (dotted line) between current location and B. Then LLA checks whether the sensing radius of all sensors belonging to next cluster intersects with the line. In Fig. 6, dotted line is checked to be a valid segment for data mule to collect all sensor data of next cluster. Obviously, it's shorter path length than the original. In another case as displayed in Fig. 7, the topology is kept the same only with different current location. LLA also conducted the check between dotted line and sensors. It will find only a sensor can be passed through if following the dotted line. LLA returns false message to notice data mule to move in the original way.

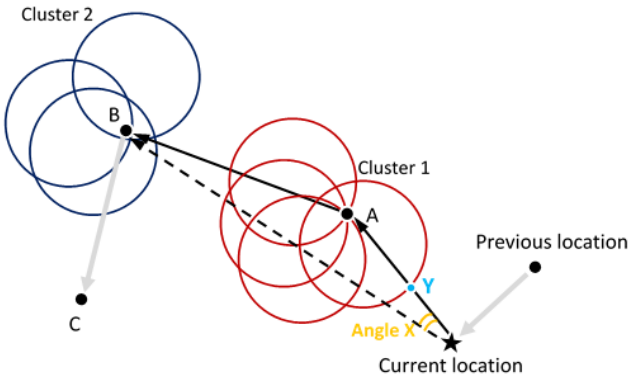


Fig. 6. Dotted line is a LLA segment with data of cluster 1 collected.

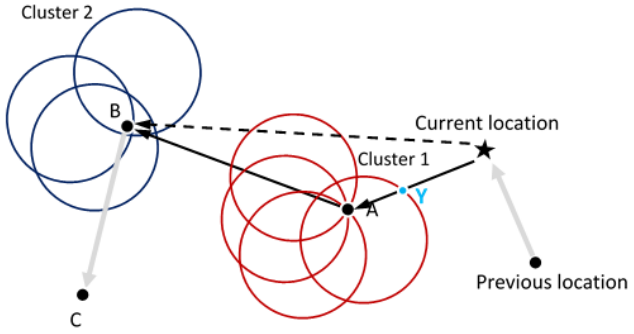


Fig. 7. Dotted line is a LLA segment with only one sensor data collected.

However, in most cases, clusters consists of few sensors and only one or two sensors are not passed through. Advanced-LLA (ALLA) is for fixing this problem. Fig. 8, shows how ALLA works. Data from the two sensors cannot be collected when data mule goes along the dotted line. Data mule makes detours to E1 and E2 sequentially, finally reaching B. E1 and E2 are the respective nearest nodes from two sensors to the dotted line. Obviously, ALLA route (current-E1-E2-B) is still shorter than the original route (current-A-B). The number of detours of data mule is a trade-off with the efficiency, and taking too many detours may cause longer path instead.

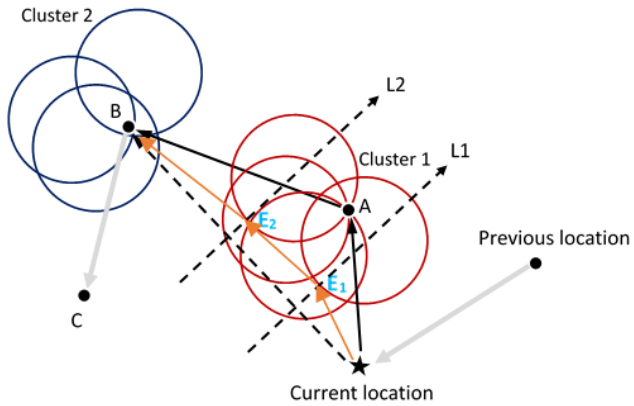


Fig. 8. ALLA with two detours to E_1 and E_2.

LLA can reduce the *Rotation Degree of Data Mule*, denoted as ***RDDM***. In Fig. 6, *RDDM* of LLA route (previous-current-B-C) is less than original route (previous-current-A-B-C) by two times of angle X. The reducing of *RDDM* not only saves the power consumption but decreases orienting errors. Moreover, LLA can lower the data communication delay while data gathering is conducting. In original way, visiting waypoints are always in common intersection area of clusters. In some ways, there is higher rates at waypoint to cause queuing delay and even data collision. In Fig. 6, following the original way, data mule can collect data of one sensor at Y, and for the remaining three sensor data it must move to A. That is, if there are sensor data not collected after reaching next waypoint (A in this example), data mule must stop to collect remaining data by turns until all data is received correctly and hence delay occurs. That takes data mule more time to complete the task. On the other hand, we define ***Max Step*** as the max segment during whole data gathering task. LLA will lead to larger *Max Step*. In this paper, for saving the computational time, we just look one ahead and ALLA makes only a detour for one missing sensor while applying LLA. But, LLA is feasible for multiple look-ahead clusters and ALLA can also make multiple detours. It depends on your needs to guide a better route for the data mule. More stimulation results and analysis are in section VI.

VI. Simulation Results and Analysis

In our studies, a C++ program was used for simulations. The environment was a fixed two-dimensional map with sensors randomly distributed. For the sake of accuracy, each simulation was tested for multiple runs to obtain the average. 2-opt is configured to test at most 10 times in a round for improvement; if no chromosomes with higher fitness arise, chromosome keeps the same. Simulations will stop when GA converges. While GA produces the same result for successive 15 generations, it converges and ***Convergence*** is defined as the total required generations. In this section, the effects of Balanced Standard Deviation Algorithm and Look-Ahead Locating Algorithm were discussed. Also, some reasonable explanations were given for them.

Each cluster was given a waypoint by CA. Then, a route visiting all clusters was generated with greedy method. The length is denoted as ***CAnCP***. Then, CP was applied on *CAnCP*; the length is denoted as ***CACP***. From the previous research, we found CP can save about 6% route length no matter how the network topology changes. In this paper, we evaluated the experimental results on the basis of how they shorten *CACP*. The performance improvement is defined as

$$\textbf{\textit{Gain}} = (CACP\text{-}\textbf{\textit{Avg}})/CACP*100\%.$$

where *Avg* is the average route length of multiple runs.

A. Effect of Balanced Standard Deviation Algorithm

In this section, we tune the parameters to manifest the effectiveness of BSDA. The number of clusters is set to 73, which is a prime number. And the size of population is an integral multiple of 72 (73-1). In this environment, SD of BSDA is always 0, but SD of CGA and RI increase linearly, as shown in Fig. 9. However, BSDA still works in general cases. Fig. 10 indicates the *Gain* with RI, CGA and BSDA being applied. BSDA outperforms CGA all the time and is better than RI in some cases. Fig. 11 presents the *Convergence* of CGA, RI and BSDA, all of which have the similar variation trend: decreases while the size of population increases. This phenomenon indicates that GA can find the optimal in less computation rounds among a large searching space. By the simulation results, CGA can generate relatively good chromosomes (with lower fitness value) in the first time but lacks of diversity; RI performs better but spends more generations (or computational time) to converge. Besides, BSDA could offer a balanced way to generate as more as possible different segments but was only a few generations faster than RI. In summary, the effect of BSDA was really close to RI. We concluded the tailor-made initialization schemes cannot perform well in the long term evolution. So, conducting BSDA during the evolution to give full play to the characteristic of high diversity may enhance the final results. This will be treated as future work.

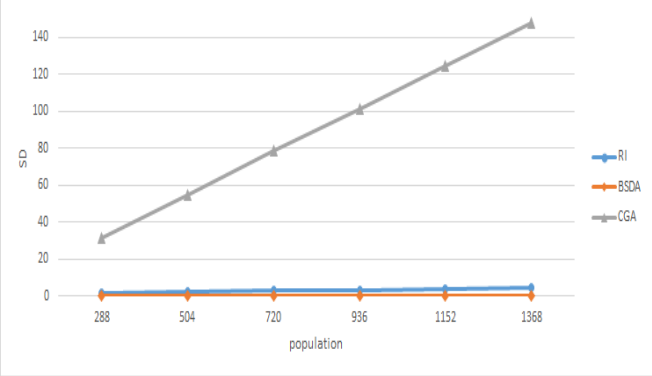


Fig. 9. Standard Deviation (average over 20 runs). map size=2000*2000, #sensor=100, sensing radius=50, #cluster=73, crossover probability=1, mutation probability=0.2, initial population generated by CGA, RI and BSDA.

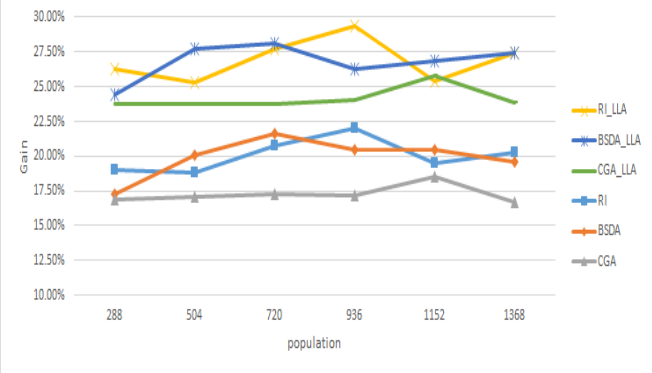


Fig. 10. Gain (average over 20 runs). map size=2000*2000, #sensor=100, sensing radius=50, #cluster=73, crossover probability=1, mutation probability=0.2, initial population generated by CGA, RI and BSDA.

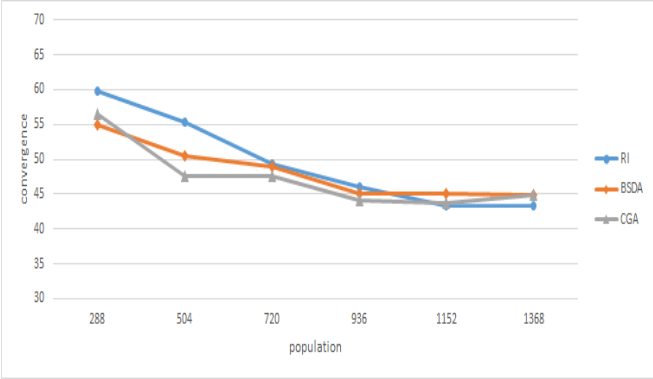


Fig. 11. Convergence (average over 20 runs). map size=2000*2000, #sensor=100, sensing radius=50, #cluster=73, crossover probability=1, mutation probability=0.2, initial population generated by CGA, RI and BSDA.

B. Effect of Look-Ahead Locating Algorithm

In this section, we discuss the effectiveness of LLA in some different aspects such as ***LLA hits*** (a LLA segment is valid), ***LLA improvement***, ***Rotation Degree of Data Mule (RDDM)***, ***Max Step*** and ***Ruggedness***. Table 2 shows the simulation results. To keep the simplicity of GA and computational efficiency, LLA is not embedded in GA but as a plugin only applied on the best chromosome after the evolution. By LLA, data mule can ignore about 39% (LLA hit rate) clusters and pass through directly during the gathering task. LLA reduces the total path length with about 6.7% on average (LLA improvement). Moreover, LLA also lowers the *RDDM* for 35% on average, about 4.5 spinning rounds (4.5x360°). Moreover, the reduction can be much more in a denser WSN. On the other hand, LLA will lead to larger segments, such as (current-B) being longer than (current-A) and (A-B) in Fig. 6. According to the simulation results, LLA causes about 21% increment of *Max Step*.

We define an index, denoted as $\boldsymbol{Ruggedness_{local}}$, to evaluate the smoothness grades of each segment.

$$\text{Ruggedness}_{local} = \text{Rot}/\text{Seg}.$$

Where ***Seg*** is the length of current segment and ***Rot*** is the rotating degree. *Rot* is always less than 180°, oriented to the next segment. As demonstrated in Fig. 12, *Seg* is the length of AB and *Rot* is 180°-∠ABC.

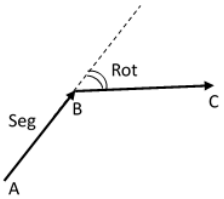


Fig. 12. Illustration of Ruggedness calculating. Ruggedness of segment{AB} is defines as Rot/Seg.

Fig. 13 and Fig. 14 exhibit the $Ruggedness_{local}$ of every segment without or with LLA. Obviously, the average with LLA is smaller than the other. The horizontal bar with different length stands for disparate segments. Nearly all the segments of LLA are longer. Also, the variation degree of $Ruggedness_{local}$ is different; a sharp rise of $Ruggedness_{local}$ may occur in Fig. 13. Then, taking it in the task view, we evaluate the whole gathering task with $\boldsymbol{Ruggedness_{global}}$.

$$\text{Ruggedness}_{global} = \sum Rot / \sum Seg.$$

According to Table 2, average $Ruggedness_{global}$ is 0.33 and 0.23 without or with LLA, respectively; that means the route guided by LLA is smoother on about 30%. To move in a smoother and shorter route, data mule not only saves power consumption but reduces orienting errors.

Table 2. Experimental results (average over 20 runs). map size=2000*2000, population=300~1300, #sensor=100, sensing radius=50, crossover probability=1, mutation probability=0.2, initial population generated by CGA, RI and BSDA.

	300			500			700		
	CGA	RI	BSDA	CGA	RI	BSDA	CGA	RI	BSDA
Gain (%)	16.85	19.27	18.20	14.83	19.95	18.11	19.92	20.53	18.76
Gain after LLA (%)	23.20	25.60	24.18	22.27	26.70	25.16	27.07	26.98	26.54
LLA improvement (%)	6.35	6.33	5.99	7.44	6.75	7.05	7.15	6.45	7.78
Max Step	666.99	568.29	571.99	689.68	535.03	573.89	628.67	509.42	523.29
Max Step after LLA	779.96	681.87	674.05	784.57	709.32	661.04	700.78	680.81	617.60
Max Step increase (%)	16.94	19.99	17.84	13.76	32.58	15.19	11.47	33.64	18.02
RDDM	4400.1	4739.0	4489.4	4423.8	4687.6	4688.9	4588.3	4610.2	4584.2
RDDM after LLA	2931.8	3225.7	2998.1	2891.9	3043.0	3049.6	2952.4	3054.6	2795.7
RDDM reduction (%)	33.37	31.93	33.22	34.63	35.08	34.96	35.66	33.74	39.01
#Cluster	74.4	75.9	74.35	73.85	74.8	74.25	74.85	75.35	74.7
#LLA hit	28.85	29.25	28.85	28.95	28.9	28.7	28.8	29	29.35
LLA hit rate (%)	38.78	38.54	38.80	39.20	38.64	38.65	38.48	38.49	39.29
Ruggedness	0.31	0.34	0.32	0.31	0.34	0.35	0.32	0.33	0.34
Ruggedness after LLA	0.23	0.25	0.23	0.22	0.24	0.25	0.23	0.24	0.23

	900			1100			1300		
	CGA	RI	BSDA	CGA	RI	BSDA	CGA	RI	BSDA
Gain (%)	17.95	20.20	20.04	17.80	19.79	20.05	17.90	20.47	19.58
Gain after LLA (%)	26.34	26.55	26.60	24.02	25.38	27.43	24.70	26.80	25.73
LLA improvement (%)	8.40	6.35	6.56	6.21	5.59	7.37	6.81	6.33	6.15
Max Step	593.85	568.26	575.09	695.52	560.01	516.89	633.05	520.46	592.49
Max Step after LLA	724.06	755.73	694.20	771.92	671.28	655.3	737.66	647.06	687.93
Max Step increase (%)	21.93	32.99	20.71	10.98	19.87	26.78	16.52	24.32	16.11
RDDM	4644.4	4650.3	4714.7	4515.4	4733.8	4604.7	4535.4	4612.1	4532.1
RDDM after LLA	2882.3	3012.3	2973.3	2992.4	3152.55	2925.4	2994.65	2973.6	2936.31
RDDM reduction (%)	37.94	35.22	36.94	33.73	33.40	36.47	33.97	35.53	35.21
#Cluster	75.85	75.7	74.65	73.85	75.3	74.4	75.15	74.4	74.1
#LLA hit	30.05	29.7	29.2	28.75	28.75	28.75	29.05	28.65	28.6
LLA hit rate (%)	39.62	39.23	39.12	38.93	38.18	38.64	38.66	38.51	38.60
Ruggedness	0.33	0.34	0.34	0.32	0.34	0.34	0.32	0.34	0.33
Ruggedness after LLA	0.23	0.24	0.24	0.23	0.25	0.24	0.23	0.24	0.23

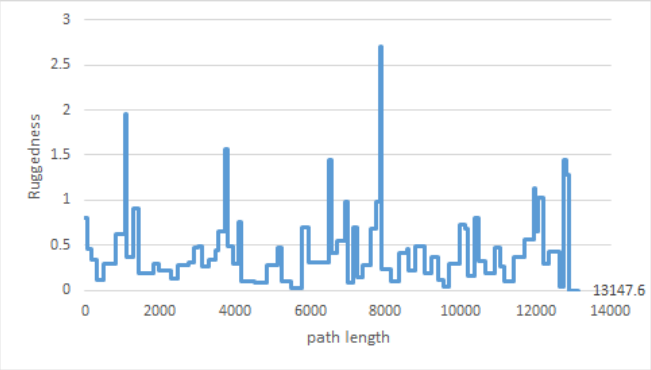


Fig. 13. Ruggedness without LLA during all path and the average segment length is 172.99. map size=2000*2000, #sensor=100, sensing radius=50, crossover probability=1, mutation probability=0.2, population=300, initial population generated by RI.

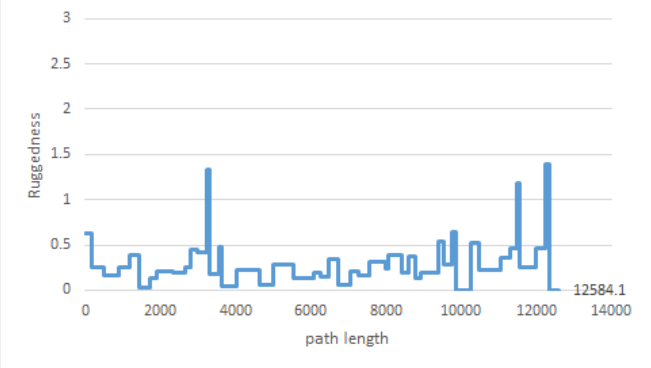


Fig. 14. Ruggedness with LLA during all path and the average segment length is 267.75. map size=2000*2000, #sensor=100, sensing radius=50, crossover probability=1, mutation probability=0.2, population=300, initial population generated by RI.

VII. Conclusions

Motivated from the characteristics of data collection using data mule in WSN, we proposed a clustering-based genetic algorithm (CBGA). In GA, Balanced Standard Deviation Algorithm (BSDA) is for generating initial population in a balanced way. Later, an approximated TSPN route will be generated while GA converges after multiple generations. We further improved the route by shortcut schemes, Look-Ahead Locating Algorithm (LLA) and Advanced-LLA (ALLA). ALLA solves the problem of missing only few sensors. Furthermore, simulation results are analyzed in terms of some aspects such as LLA hits, LLA improvement, *RDDM*, *Max Step* and *Ruggedness*. Besides, CA is beneficial to the use of LLA because CA can check the possible entry points in the common intersection area of each cluster for shortcuts to increase the LLA hits. Combined with CBGA, LLA offers a further 6.7% improvement of *Gain*. We also depict Fig. 13 and Fig. 14 to show the smoothness grades of a TPSN route without or with LLA. In the view of whole task, $Ruggedness_{global}$ can be reduced about 30%. Finally, it was noted that our solution is practical and applicable to other problem domains such as the tour planning of computer-vision based inspection (uses a set of end-effector placements for taking pictures [2]). Moreover, GA is modularized and some features can be extended easily as plugins.

References

[1] E. M. Arkin and R. Hassin, "Approximation algorithms for the geometric covering salesman problem," Discrete Applied Mathematics 55 , no. 3, 197–218, 1994.

[2] I. Gentilini, F. Margot and K. Shimada, "The travelling salesman problem with neighborhoods: MINLP solution," Optimization Methods and Software, 28(2), pp.364-378, 2013.

[3] L. He, Jianping Pan and Jingdong Xu. "A progressive approach to reducing data collection latency in wireless sensor networks with mobile elements," IEEE Transactions on Mobile Computing, vol. 12, issue. 7, pp.1308-1320, 2013.

[4] P. N. Pathirana, T. J. Black and S. Nahavandi, "Path planning for sensor data collecting mobile robot," In Intelligent Sensors, Sensor Networks and Information Processing Conference, 2005.

[5] D. Kim, R. N. Uma, B. H. Abay, W. Wu, W. Wang and A. O. Tokuta, "Minimum Latency Multiple Data MULE Trajectory Planning in Wireless Sensor Networks," IEEE Transactions on Mobile Computing, 2013.

[6] A. Wichmann, J. Chester and T. Korkmaz, "Smooth path construction for data mule tours in wireless sensor networks," In Global Communications Conference (GLOBECOM), pp.86-92, December 2012.

[7] M. Ma and Y. Yang, "Data gathering in wireless sensor networks with mobile collectors," IEEE International Symposium on Parallel and Distributed Processing, pp.1-9, 2008.

[8] Y. C. Tseng, F. J. Wu, and W. T. Lai, "Opportunistic data collection for disconnected wireless sensor networks by mobile mules," Ad Hoc Networks, vol. 11, pp.1150-1164, 2013.

[9] M. Dunbanin and L. Marques, "Robotics for environmental monitoring," IEEE Robotics and Automation Magzine, pp.24-39, March 2012.

[10] J. S. Liu, S. Y. Wu and K. M. Chiu, "Path Planning of a Data Mule in Wireless Sensor Network Using an Improved Implementation of Clustering-Based Genetic Algorithm," 2013 IEEE Symposium Series on Computational Intelligence, Singapore.

[11] J. H. Holland, "Adaptation in Natural and Artificial Systems," Ann Arbor, MI: The University of Michigan Press, 1975.

[12] D. Simon, "Evolutionary Optimization Algorithms," John Wiley, 2013.

[13] Z. H. Ahmed, "Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator," International Journal of Biometrics and Bioinformatics, vol. 3, no.6, pp.96-105, 2010.

[14] S. M. Abdel-Moetty and A. O. Heakil, "Enhanced traveling salesman problem soving using genetic algorithm technique with modified sequential constructive crossover operator," International Journal of Computer Science and Network Security, vol. 12, no. 6, June 2012.

[15] G. A. CROES, "A method for solving traveling salesman problems," Operations Res. 6 , pp.791-812, 1958.

[16] I. M. Oliver, D. J. Smith and J. R. C. Holland, "A study of permutation crossover operators on the traveling salesman problem," In Proceedings of the second international conference. on genetic algorithms (ICGA'87) (pp.224–230). Cambridge, MA:Massachusetts Institute of Technology, 1987.

[17] K. Katayama, H. Hirabayashi and H. Narihisa, "Performance analysis of a new genetic crossover for the traveling salesman problem," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E81-A No.5, pp.738-750, 1998.