Robot Routing Using Clustering-Based Parallel Genetic Algorithm with Migration

Ko-Ming Chiu, Jing-Sin Liu*, Shih-Rong Yang

Institute of Information Science, Academia Sinica, Taipei, Taiwan 115, ROC

*liu@iis.sinica.edu.tw

***Abstract*- Wireless sensor network (WSN) technology is widely used in environment monitoring, health care, surveillance systems and unmanned space or planet exploration. To collect the exploration data, controllable mobility such as mobile robot called data mules can be used to help complete data gathering missions using the same sensors. This study focuses on data gathering by a mobile robot in a WSN, also referred to as a robot routing problem. It is formulated as a Traveling Salesman Problem with Neighborhoods (TSPN). Basing on the need of practical applications, a two-phase method, a clustering-based parallel genetic algorithm with migration (CBPGA), is proposed as a TSPN route computing and optimizing scheme that allows simultaneously searching of the order and locations of waypoints of clusters. The first phase of the method consists of a Euclidean-distance based clustering algorithm that assigns the nodes to clusters, and the second phase routes these clusters by planning the data collection points in the common intersection area of each cluster and the order to visit them using a genetic algorithm (GA). A travel cost reduction scheme by finding the closest waypoint in a cluster according to the visiting order is used to further shorten the path length. Simulation studies are conducted to evaluate the performance of CBPGA, by comparing it with other solutions to TSPN in WSN with identical or random sensing radii. To reveal the relative performance of each solution scheme, the effect of clustering, the role of GA and parallelism, and the integration of CBPGA in data-gathering route design are highlighted.**

Keywords- Data Collection; Traveling Salesman Problem with Neighborhoods (TSPN); Clustering; Genetic Algorithm

I. INTRODUCTION

Rapid advances in the technology of wireless sensor networks (WSN) have led to its incorporation in a variety of applications by deploying small and inexpensive sensors, such as environment monitoring, surveillance systems, disaster rescue, underwater and unmanned space or planet exploration. In unmanned planet exploration, sensors play an important role; a number of sensors are often distributed over a geographic area with the capacity to communicate with each other across limited distances. As a result, numerous studies focus on gathering data by sensor networks [1-4]. There are multiple ways to collect the data in a wireless sensor network. A common approach is to relay the data in each sensor node to a data sink through multi-hop communication. Energy restrictions prevent a sensor's communication range from increasing limitlessly until all sensors or a monitored region is covered, while multi-hop data forwarding consumes a large amount of energy. Consequently, in networks where sensors are sparsely deployed, sensors may not be fully connected, and some nodes may be used on many paths, depleting their energies earlier due to load unbalance.

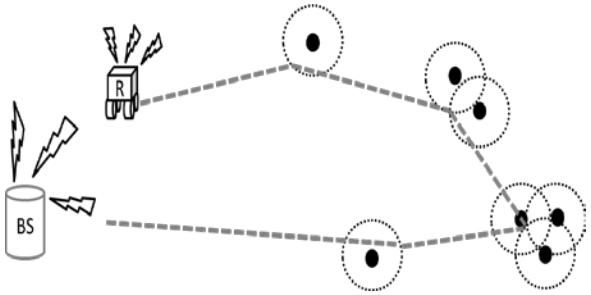


Fig. 1 Navigating a mobile robot (R) along a dashed-line path to download data from or communicate with other sensors and back to base station (BS), a data sink responsible for transferring the collected data to user through gateway

To transfer data stored in nodes to base station without suffering the common pitfalls of the centralized scheme mentioned above, alternatively one promising and practical solution for this data collection problem is to employ one or more mobile robots/elements mounted with sensor and wireless communication devices with powerful memory and energy, called data mules (also referred to as mobile elements, mobile sinks, or message ferries synonymously in literatures). The data mules can communicate with sensor nodes to automatically gather data from all sensors and may prevent the sink area to become a bottleneck. The data mules can follow random, predictable, or controlled mobility to move close enough to the sensor nodes to gather data through one-hop communication. In general, adding mobility helps to support disparate missions using a common set of sensors. Therefore, to jointly accommodate sensing and routing, issues, such as scheduling, path planning and speed control, related to exploiting one or multiple data mules for collecting data in sensor networks become important [5-8] (see Fig. 1). Recent studies [9-11] have shown that for nodes with different densities, the use of a data mule indeed significantly reduces data latency and energy consumption at sensor nodes compared with the commonly-used multi-hop forwarding approach, thus

prolonging network lifetime [3]. However, a fatal drawback due to the use of a data mule is the increase of data delivery latency in WSN. In order to decrease such latency, the traveling path of a data mule must be shortened. Thus, effective use of a data mule for collecting data in wireless sensor networks involves generating a path along which the mobile robot can retrieve data from sensors in a field while minimizing overall travel costs (related to latency of data delivery) is subject to hard or soft constraints on path or network, which allows a integer linear programming (ILP) formulation of data collection problem [10].

Robot routing problem [1-3, 10, 12] is a tour construction problem. It involves generating a path along which the mobile robot can retrieve all data from all sensors while minimizing overall travel costs (distance). Generally, the robot routing problem can be regarded as a special form of the Traveling Salesman Problem with Neighborhoods (TSPN) [12, 13], a variant of the Traveling Salesman Problem (TSP) known as the NP-hard problem [14, 15], where each neighborhood is a disk region whose center corresponds to a node and the radius is determined by the communication range. Although TSP has been long and extensively studied [15], TSPN solutions were first studied by Arkin and Hassin [13] to examine a collection of several regions in a plane, called neighborhoods, and find the shortest tour to visit all neighborhoods. Solutions to TSPN, such as the greedy method, the approximation algorithm in [16], or the Genetic Algorithms (GAs) presented in [17], are often based on heuristically optimizing a TSP route. These methods do not effectively take advantage of the sensing ranges of sensor nodes, thus can't effectively decrease travel costs for a WSN containing sensors with overlapping communication ranges. The mobile robot needs only to visit a specific location within the effective communication range of a sensor to download data, instead of visiting the sensor's precise location.

A variety of data gathering schemes using data mule have been proposed in recent years. Yuan et al. [12] presented a data-gathering route design method for robot routing problem. However, they did not consider situations involving sensors with overlapping communication ranges. [4] partitioned the network into a collection of covering sets so that each node belongs to at least one set. Each set has one polling point for the robot to download all data in the set using one-hop communication. The route is the solution to TSP of the polling points. RaceTrack [6] was proposed to decrease travel costs of an initially constructed non-intersecting TSP route in a step-by-step manner by exploiting shortcut heuristics. RaceTrack first constructs a TSP route without self-intersecting from the sensor nodes' locations. Then inner lane substitute points, concave bend substitute points and shortcut procedure are used for route optimization. The mobile robot can choose a location to visit within the effective communication range of a sensor to download data, and thus does not need a specific location.

According to previous studies about clustering [18] motivated by scalability and efficiency of the sensor network, the mechanism of clustering is indeed beneficial to the reduction of data delivery latency in wireless sensor networks. On the other hand, the advantage of GA-based approach for route optimization is generally recognized as a better scalability to higher number of nodes and a spectrum of possible routes are evaluated via evolution toward near optimal routes. By taking advantage of overlapping sensing ranges and noting that the number and locations of waypoints and the order to visit them that a path traverses is a contributing factor to path length, this paper proposes a clustering-based parallel genetic algorithm with migration (CBPGA). The algorithm decreases the overall necessary number of nodes to visit in the route and simultaneously find locations of waypoints and the order to visit them to form a shorter data-gathering route.

The remainder of this paper is organized as follows. After examining related researches in section II, the model and problem definition are described in section III. Section IV introduces in detail the clustering-based parallel genetic algorithm with migration. Section V presents the comparative simulation results of CBPGA and alternative TSPN solutions on which the observations and analyses are based. Finally, section VI concludes the paper. Throughout the paper, tour, path and chromosome are used interchangeably; so are data mule and mobile robot.

II. MODEL AND PROBLEM DEFINITION

A. Network Model

The network is composed of a set of sensors (nodes) {s1, s2, s3, … , sn} which are data sources randomly distributed over a two-dimensional plane, and one mobile robot for carrying out data collection by traversing the whole network. It is assumed assumed that each sensor si can sense a disc region centered at si with radius rs. Each sensor is equipped with omnidirectional antennae to communicate with other sensors or the mobile robot within the disc region. It is assumed that the mobile robot (R) has unlimited resources (memory, battery…) to traverse the whole network for visiting the nodes to download data or communicate with other sensors when the mobile robot is within their transmission ranges. In this paper, the sensing range (rs) and transmission range (rt) of each sensor are the same.

B. Problem

The robot routing problem involves a mobile robot moving from a given start position (S) to download data from each sensor within communication range (rs) in a prior-deployed static wireless sensor network. The mobile robot returns to the start position after finishing the data-gathering task. The main objective (e.g. [2, 3, 9, 10]) is to plan a proper order, usually with a total path length as short as possible, for the robot to visit waypoints of all clusters and gather all data from every sensor.

It is noticed that there are distinctions between symmetric TSP and TSPN. Each node in TSP is visited only once and a TSP

route is completely determined by its ordering of nodes where the distance between two nodes is independent of the ordering of nodes. On the contrary, the neighborhood of a node in TSPN is an area where the waypoints can be. Hence, the order of cluster visiting, the number of waypoints, and the precise locations of waypoints are all contributing factors to path length in a data-gathering task conducted by the data mule. In order to decrease traveling costs and enhance the data gathering efficiency, the core focus of this study is to reduce the number of visited nodes while ensuring the robot can download all data from all sensors, and to provide the robot a possibly short path between nodes by selecting the waypoint and simultaneously deciding the order to visit them. These two concerns represent the heart of this research, and a clustering-based parallel genetic algorithm with migration (CBPGA) is proposed as the joint solution. The solution is a one-hop data gathering scheme; as the mobile robot travels along the planned tour, the data gathering is executed cluster-by-cluster.

III. CLUSTERING-BASED PARALLEL GENETIC ALGORITHM WITH MIGRATION

Based on Darwin's principle of natural selection –survival of fittest, Genetic Algorithms (GAs) [17], one of metaheuristic algorithms, is a powerful method of heuristic random search and optimization. GAs are widely used to effectively discover near-optimal solutions to a variety of NP problems, e.g. TSP, scheduling, and shortest path routing within a reasonable amount of time. In particular, there have been many GA-based methods to TSP that employ delicate designs on top of genetic operators for finding interesting solutions [15]. GAs are implicitly parallel and however, when they are applied to more complicated problems or large-scale problems (e.g. the number of cities increases in TSP), the time required to find an adequate solution increases and may get trapped in the local optima. Both matters make it hard for GAs to obtain solutions of quality. Therefore, researchers have looked for multiple ways by which to speed up GAs. In these ways, the parallel architecture [19], called parallel genetic algorithms (PGAs), is commonly used to improve the convergence rate of GAs, instead of a long single run. A population is divided into a number of subpopulations and a specific task assigned to each task. There are three major types of PGAs that differ in the way of communication and information exchange among the subpopulations: (1) master-slave model, (2) coarse-grained model (island model) and (3) fine-grained model (cellular model). More related details of PGA are introduced in [19-21].

A. Master-Slave Parallel Genetic Algorithms

In this paper, the structure of master-slave model is adopted to achieve higher diversity of population and escape from being trapped in local optima, in addition to realize parallel processing. In the master-slave model, one computing node is designated as the master while the others are designated as slaves, which are separated from each other and evolve independently. The master will assign one or more evolutionary tasks (i.e. the most time-consuming fitness evaluation, crossover and mutation) to the slaves. This is done by distributing the whole population into subpopulations to the slaves for performing the standard GA process, and after each generation the master will receive the best result from the slave nodes. To ensure that successful solutions can be spread to other nodes, the nodes will occasionally exchange chromosomes with each other, called migration.

B. Migration Strategy

In the master-slave parallel genetic algorithm, migration operation is the most common method to provide a mechanism of occasional genetic information exchange between slaves [22, 23]. There are five migration parameters: (1) migration frequency: how frequently migration occurs, (2) migration size/rate: how many individuals migrate, (3) migration scheme: which individuals are migrated and which individuals are replaced, and (4) migration topology: [20]. Many researches on the convergence rate of PGAs focus on how to configure appropriate migration parameters. According to observations of effects of these parameters to evolutionary performance [19, 20, 22], an appropriate migration configuration can help obtain both better and global results within fewer generations, thus accelerating the convergence. By this mechanism, a slave can share their own best chromosome with other slaves in the parallel genetic algorithm. Existing researches have found the results using migration significantly better than those without migration.

C. Overview of CBPGA

To accelerate the convergence rate of generating the near-optimal shortest route, the structure of the master-slave PGA with migration [9, 19-23] incorporating with the clustering algorithm (CA), travel cost reduction scheme (RS) are combined to form CBPGA. Aiming to generate a collection of near-optimal solutions to robot routing problem, CBPGA will plan a near-optimal route to be traversed by the mobile robot.

The basic components of CBPGA are depicted in Fig. 2. The main tasks of the CBPGA master node include executing the clustering algorithm, creating the initial population by chromosome generation algorithm (CGA, detailed in subsequent session). When a slave receives a subpopulation from the master, the slave performs the whole evolutionary process (i.e. selection operation, crossover operation, mutation operation and migration) independently and synchronously until the terminal condition is satisfied. In each slave node, the migration operation is performed periodically. A slave only returns the best result to the master when the evolutionary process is terminated. When the master receives the results from all slaves, it selects the best result among these received results as the output result. For a given valid chromosome, the route is found by a closest

waypoint (CP) finding algorithm in each cluster to be visited from an immediate past visited point, and is used to further reduce traveling costs. These processes are introduced in the following sections.

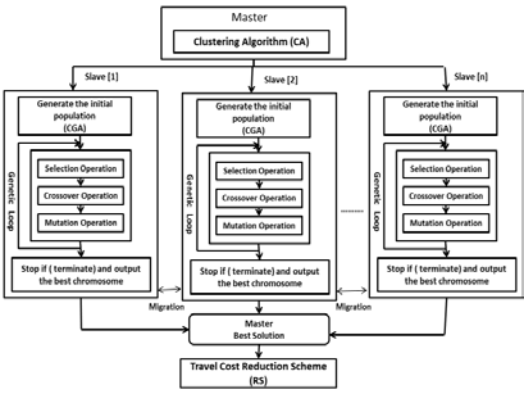


Fig. 2 The structure of the clustering-based parallel genetic algorithm with migration (CBPGA) for robot routing

D. Clustering Algorithm (CA)

In order to decrease overall traveling distance that a data mule must traverse in a field, a core focus of our study is to reduce the number of visited nodes while ensuring the data mule can precisely collect the information gathered by a sensor network, and this is achieved through clustering algorithm (CA) in this paper. The result of CA is dispatching each sensor node into a certain cluster, thus dividing the global network into small clusters to reduce the overall computing complexity. The data mule travels only to the visited point in intersection areas of neighboring nodes classified as the same cluster to collect the data of all sensors using one-hop communication. This navigation makes all wireless communication between the sensing nodes and the data mule short range, so that the data mule could follow a shorter route to finish the data collection task in sensor network: the nodes could send their data directly to the data mule as the data mule visits a specific location within the effective communication range of a sensor (or a cluster), thus saving the time and energy. However, the sensing range of a sensor node is affected and adjusted by its power level. It should be noted that the sensing ranges affect the clustering of sensors and the path of a data mule. Therefore, a clustering algorithm is proposed in this paper to handle the classification of sensors with variable sensing ranges.

Before introducing CA, it is assumed that all nodes are stationary and have an ID. Two correlated tables are created (shown jointly as Table 1) offline to record the relationships between nodes and clusters for quick reference to the clustering algorithm. Each table entry corresponds to a cluster ID and one or more node IDs. The Node-Cluster Table has two columns (Node ID vs. Cluster ID), where each node is matched with its corresponding clusters and initial values are all set to null. The reciprocal is true of the Cluster-Node Table.

TABLE 1 TWO TABLES SHOW THE RELATIONSHIP BETWEEN NODES AND CLUSTERS (LEFT: NODE-CLUSTER TABLE. RIGHT: CLUSTER-NODES TABLE)

Node ID	Cluster ID	Cluster ID	Node IDs
Node 1	Cluster 1	Cluster 1	Node 1
Node 2	Cluster 2	Cluster 2	Node 2, Node 4
……	……	……	……
Node n	Cluster m	Cluster m	Node n

Pseudo-code of the clustering algorithm is described as follows.

Clustering algorithm (CA): The clustering algorithm classifies all sensors with variable sensing ranges into disjoint clusters. Sensors belonging to the same cluster have a common intersection area.

Input: a set of sensors $\{s_1, s_2, s_3, \ldots, s_n\}$ and sensing radius (communication range) r_i, $r_i, 1 \le i \le n$ of each sensor.

Output: a set of disjoint clusters $L = \{L_1, L_2, \ldots, L_m\}$.

- Initially, $L = \Phi$ and the corresponding Cluster IDs of all sensors are set to *null*.
- For each sensor pair (s_i, s_j), $i, j \in \{1, 2, \ldots, n\}$ and $I \neq j$. If Euclideandistance$(s_i, s_j) \le r_i + r_j$, go to step 3. Else, go to step 2. The function Euclideandistance(p, q) is defined as the Euclidean distance between two points p and q.

- If s_i and s_j don't have the common Cluster IDs, go to step 4. Else, go to step 10.
- If cluster IDs of s_i and s_j are *null*, go to step 5. Else, go to step 6.
- If Euclideandistance(s_i, s_j) $\leq r_i + r_j$, assign the cluster IDs of s_i and s_j to a new cluster ID and go to step 10. Else, assign the cluster IDs of s_i and s_j to a new cluster ID respectively and go to step 10.
- If cluster IDs of s_i are not *null* and cluster IDs of s_j are *null*, go to step 7. Else, go to step 8.
- Assign the cluster ID of s_j to cluster ID of s_i when Euclideandistance(s_p, s_j) $\leq$ ($r_p + r_j$), s_p Є the cluster of s_i and go to step 10. Else, assign the cluster ID of s_j to a new cluster ID and go to step 10.
- If cluster IDs of s_i and s_j are not *null*, go to step 9. Else, go to step 10.
- If Euclideandistance(s_p, s_i) $\leq$ ($r_p + r_i$), s_p Є the cluster of s_j and Euclideandistance(s_q, s_j) $\leq$ ($r_q + r_j$), s_q Є the cluster of s_i, add the cluster ID of s_j to the cluster ID of s_i and go to step 10. Else, directly go to step 10.
- If all sensors are divided into a set of clusters *L*, return with *L*. Else, go to step 2.

Through the above clustering algorithm, a sensor network consisting of sensors with variable sensing ranges is divided into a set of clusters L = {L1, L2, ……, Lm}. The sensors belonging to a certain cluster have the same intersection area. In sparse sensor distributional environment, there may exist many clusters that only include one sensor. Fig. 3 shows an example where CA classifies several sensors that share a common intersection area to the same cluster. For this example, one sensor does not have a common intersection area with any other sensors and one sensor does not have a common intersection area with all sensors of Cluster-2. Therefore, these two sensors are classified into two new clusters, Cluster-3 and Cluster-4, respectively.

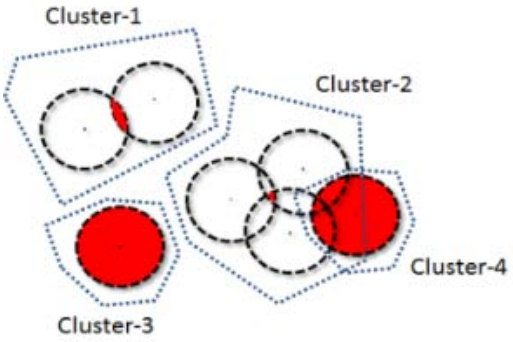


Fig. 3 The partition of seven sensor nodes into four clusters.

E. Travel Cost Reduction Scheme (RS)- Finding the Closest Waypoint (CP) for Each Cluster to a Given Point

Assume that all nodes are clustered and the data in a cluster could be gathered by only one waypoint visiting. In order to further reduce robot travel costs, a heuristic that works well should find the closest waypoint among a finite number of candidate waypoints for the next-visiting cluster from the just-visited waypoint, called the travel cost reduction scheme (RS). However, the genes of each chromosome consist of clusters, and depending on what order they appear in the chromosome, the closest waypoint varies. Following the ordering of the chromosome, as the data mule moves to a location S, it selects the next data-gathering site as the closest position of the target cluster by the following algorithm.

RS: This algorithm finds the closest position (CP) in a target cluster from a location as a waypoint, depending on the order of visit.

Input: Clusters generated by CA, start position S representing the immediate recently visited point, and next target cluster to visit.

Output: the coordinate of the closest waypoint within the target cluster from S.

- Find all sensors belonging to a target cluster using CA.
- Calculate the points of intersection (circle-circle intersections) based on communication ranges of the sensors in the target cluster and add these points to a set of points (PC).
- Calculate cross-points from S to the center of each sensor (line-circle intersections) and add these points to PC.
- Select from PC a point with shorter Euclidean distance to S. This point is the closest waypoint, and the data mule can move to this point to collect all the data of the sensors in this target cluster.

Fig. 4 shows an illustration of CP scheme in which ***S*** denotes the just-visited waypoints and a target cluster with three sensors. In step 2 of CP algorithm, three points (A, B and C) of intersection can be obtained. In step 3, three cross-points (D, E and F) are calculated. Thus, PC = {A, B, C, D, E, F} and |PC| = 6. Finally, point B will be selected as the closest position (CP), because the data mule can collect data of all sensors in the target cluster at point B and the distance between S and B is the shortest. Note that the waypoint in a target cluster is varied according to the location of the least recently visited location of the mobile robot, and thus is varied according to the absolute visiting order of the data mule. This is important to TSPN route

design, in contrast to TSP, which is often symmetrical so that the route depends only on the relative order of nodes.

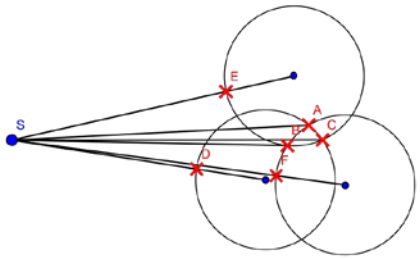


Fig. 4 Illustration of finding the closest waypoint for a cluster to a given start point S representing the immediate recently visited point. Positions A-F on the boundary of sensing ranges are candidate points.

IV. IMPLEMENTATION OF PARALLEL GENETIC ALGORITHM

There are many different ways to implement a PGA. In practice, minor variation in implementation details could yield different achievable performance levels. The implementation of PGA in this paper for finding a shorter route of a mobile robot in WSN is detailed as follows. As genetic operations often destroy the feasibility of chromosomes, only valid chromosomes are retained to improve the implementation efficiency.

A. Chromosome Encoding

The set whose permutations are provided by the visited cluster set is encoded as chromosomes in the parallel genetic algorithm to explore a spectrum of routes via evolution toward the near-optimal solution. Because a mobile robot has to return to its start position to transmit data to the base station after completing its data-gathering tasks, a chromosome is represented by an ordered sequence of visited nodes with the first and last genes as start position and each gene represents a cluster ID. A chromosome with any repeated visited nodes represents a route containing a cycle and is invalid and hence, need to be eliminated. Each valid chromosome represents a feasible ordering of clusters, and each visited node is allowed to appear only once except for the first and last genes.

B. Initial Population

A population is composed of a user-specified number of chromosomes. Generating initial population with high diversity between chromosomes leads to a higher chance of evolving better offspring. In this paper, a Chromosome Generation Algorithm (CGA) is proposed based on a permutation tree to create a sufficient number of random yet feasible starting tours (non-repeated feasible chromosomes). The algorithm is as follows.

Chromosome Generation Algorithm (CGA): Chromosome Generation Algorithm is used to generate a number of non-repeated chromosomes.

Input: A feasible route (P) and population size (P_{Len}).
Output: non-repeated chromosomes in descending order of route lengths.
//*Initial Configuration– Start*
$P = S\ [S_1, S_2, S_3, \ldots\ldots, S_n]\ S$;
$N_e = \{S_1, S_2, S_3, \ldots\ldots, S_n\}$;
N_{Len} = Length of $\{S_1, S_2, S_3, \ldots\ldots, S_n\}$;
if ($P_{Len} > N_{Len}!$)
{
$P_{Len} = N_{Len}!$;
}

A number (N) can be obtained and $\min(\lfloor N! \rfloor) \geq P_{Len}$;

Randomly choose a position set $P_s = \{p_1, p_2, p_3 \ldots, p_N\}$;
Then, get the corresponding nodes $E_s = \{E_1, E_2, E_3 \ldots, E_N\}$ of N_e;
//*Initial Configuration– End*
//*Variables Declaration – Start*
ListOfPermutation = ¢;
Permutation = new int[N];
level = -1;
//*Variables Declaration - End*
{
if (*Permutation*[i] == 0)

```
                {
                    GeneratingPermutationTree(i);
                }
            }
        }
        level -= 1;
        Permutation [k] = 0;
        }
        //Function Declaration - End
        //Pseudo-Code – Start
        GeneratingPermutationTree(0);
        NewPopulation = GeneratingNewPopuationAndCalEuclideanDistance (S, Ps, Ne, PLen);
```

The initial chromosomes can be chosen randomly from the non-repeated chromosomes generated by the CGA. If the population size is larger than the total number of non-repeated chromosomes, the population size is set to the number of non-repeated chromosomes to ensure that these chosen solution combinations (chromosomes) are not identical. On the other hand, if the population size is significantly smaller than the number of non-repeated chromosomes, it is a waste to generate all non-repeated chromosomes. In order to avoid generating all non-repeated chromosomes, a number (N) of genes are first selected from the length of chromosomes so that the number of generated non-repeated chromosomes is larger than or equal to the population size (i.e $\min([N!]) \geq$ population size). Take N=10 for example. Total number of N!= 3628800 of non-repeated permutations will be generated. Then, N visited nodes are randomly selected from the visited nodes set, and their positions are recorded in the set. The illustration of CGA is briefly shown in Fig. 5. Several locations of a chromosome are selected to be permutated to produce a new permutation. The old corresponding genes are replaced by new genes in corresponding locations from a new permutation. Through these processes, we can obtain required number of non-repeated chromosomes that only differ at a number of predetermined positions, without generating all non-repeated chromosomes, thus decreasing overhead.

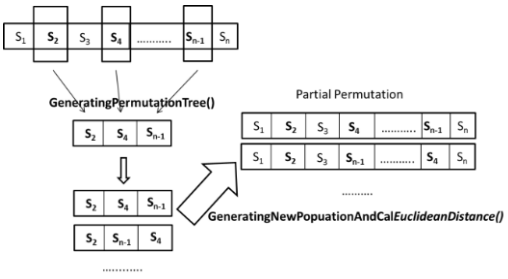


Fig. 5 The illustration of Chromosome Generation Algorithm (CGA)

When a valid permutation is given, a route is generated by the following steps.

1. Take the start position (S) of the mobile robot as the first position.
2. Take the waypoint of the closest cluster generated by RS as the next node to be visited. Else, go to step 3.
3. If all clusters have been visited, go to step 4. Else, go to step 2.
4. Take the start position (S) of the mobile robot as the last position.

C. Fitness Function

Each chromosome in the population represents a possible path generated from an ordered sequence of visited clusters. Each is evaluated by a fitness value that indicates the quality of the path. Let the fitness value of i-th chromosome be defined as the inverse of path length

$$f_i = 1/C_i$$

where Ci denotes the total cost of the i-th chromosome (permutation)

$$C_i = \Sigma C_{jk}$$

and the cost C_{jk} denotes the Euclidean distance between waypoint j and waypoint k of route i, and $C_{jk} =$ distance between waypoint j and k. Note that $C_{jk} \neq C_{kj}$, thus this is an asymmetric problem. A higher fitness value is given for a shorter route (better chromosome).

D. Selection Operation

In order to generate the next generation of chromosomes, a mating pool that consists of all the chromosomes of the current population is created. The chromosomes in the mating pool are subjected to genetic operations such as crossover, mutation,

and migration. Two parents are selected by the selection operation. And the selection operation allows the chromosomes with better fitness values to reproduce new chromosomes through the crossover operation. Parents with good fitness values are expected to produce children with better fitness values. In this paper, the chromosome with a higher fitness value between two randomly chosen chromosomes in the mating pool will be selected as a parent. To ensure the parents are valid (i.e. parents must be different), the selection operation is repeated until two valid parents are chosen.

E. Crossover Operation

The crossover operation is a probabilistic process that exchanges information between two better parent chromosomes for generating two on average even better offspring. In order to perform the crossover operation, two parent chromosomes in the mating pool will be selected by selection operation. In this paper, the crossover point is randomly chosen according to the number of visited nodes. (i.e. the first and the last gene can't be chosen as the crossover point.) For example,

Parent chromosome 1: S [A B C D E F G] S
Parent chromosome 2: S [C A B D G F E] S

Suppose the crossover point is site 3. The crossover operation will exchange the first portion of Parent chromosome 1 with the second portion of Parent chromosome 2 and vice versa. Then, the child chromosomes will be generated as follows.

Child chromosome 1: *S [A B C | D G F E] S*
Child chromosome 2: *S [C A B | D E F G] S*

The two new child chromosomes generated are subjected to the constraint of valid chromosome. Invalid child chromosomes will be discarded and valid child chromosomes will be retained. Valid new chromosome will be placed to appropriate location and the worst chromosome will be discarded. Using this approach, it is ensured that the evolution is towards shorter routes in sequential runs.

F. Mutation Operation

The mutation operation is used to increase diversity of the population to prevent premature convergence by introducing new genetic material to enlarge the search space of GA. In this implementation, each chromosome produced by the crossover operation has a small chance to be mutated based on a fixed mutation probability. Exchange mutation (i.e. exchanging two genes) is employed as the mutation operation. For each chromosome chosen to be mutated, two mutation points will be selected randomly with equal probability. Then, genes of two mutation points in the original chromosome are swapped to form a new chromosome. For example,

Original chromosome: *S [A B C D E F G] S*

If the mutation points are selected as 3 and 7, the chromosome after two-point mutation is as follows.

Mutated chromosome: *S [A B G D E F C] S*

If the mutated chromosome is valid, it is placed to appropriate location in the population and the worst chromosome is discarded. Otherwise, the mutated chromosome is discarded.

G. Migration operation

In this paper, probabilistic migration is used to speed up the spread of the currently fittest chromosome to all slaves. First a migration pool is created for information exchange between slaves. When a slave satisfies the migration condition (i.e. a randomly chosen slave executes the migration operation with migration probability periodically), it will send the best chromosome to the migration pool. When the migration pool receives the chromosome, compare it with the original chromosome in the pool. If the newly generated chromosome is better than the original, the original chromosome will be replaced with the new one. This new chromosome is then broadcasted to a slave chosen randomly. When a slave receives the new chromosome, the new chromosome will be placed to an appropriate location and the worst chromosome will be discarded.

V. SIMULATION RESULTS

The purpose of this section is to investigate the extent to which the techniques employed in CBPGA can shorten the path for robot routing problem, by comparing CBPGA with alternative solutions to TSPN. In the simulations, the sensors are randomly deployed over a 2-dimensional map of the size 2000*2000, and the number of sensors on the map ranges from a small network of the size 50 to a large network of the size 1000. In total, 50 runs are executed to get the statistics of simulation results. Each value of figures or tables is the average of 50 samples. The performance of CBPGA is presented in two parts. One is the comparison between the length and number of visited nodes of CBGA route and those of the non-clustered TSPN solutions, to highlight the effect of clustering and the role of GA in route design. The other is the validation of the effect of parallelism and migration in CBPGA.

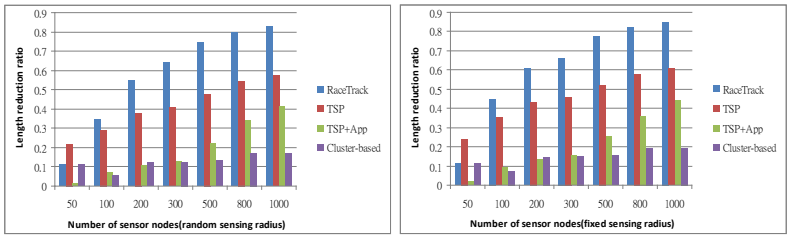
A. Part 1: CBGA (without Parallelism)

This subsection presents results of the simulations of different route design methods: greedy method, approximation algorithm [16], clustering (CA+RS), RaceTrack [4] and CBGA, to validate the advantage of integrating clustering and GA. In

the following simulations, the parameters of genetic operators are set as: crossover probability = mutation probability = 0.2 and the population size is 1000. Several factors, such as sensing radii of sensors and the distribution of sensors, affect the clustering results, since the clustering is based on proximity and sensing range of neighboring nodes. Thus, the simulations are performed for two types of networks to evaluate the impact of different, distributions of sensor nodes and sensing ranges on travel cost, in order to test and verify clustering-based genetic algorithm. The two types of networks are network where every sensor's communication range is identical and fixed at 50 and network where each sensor's communication range ranges from 20 to 50.

Firstly, the relative route length reduction of each solution to TSPN is measured as compared with CBGA route. In Fig. 4 (a) (b), the reduction rate of path length is defined as length reduction rate= (compared path length-CBPGA path length)/ compared path length for each compared route as compared to CBGA.

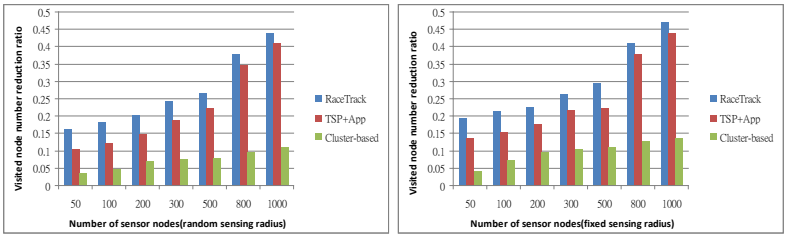
This rate indicates how far a path is from the CBGA-generated path. From Fig. 6, it is observed that the average performance of CBGA varies little for either fixed or random sensing radii, and the length reduction rate increases with the number of nodes. Performance of approximation algorithm [16] is close to clustering for medium-size network. The performance of RaceTrack is satisfactory in small networks, but degrades significantly in medium- and large-size networks. By searching for a nearly optimal ordering of the set of visited nodes provided by clustering, employing GA further optimizes the distance the mobile robot had to travel by about 10%. Also, larger length reduction rate comes up while the network size, the sensing range or the distribution of sensor nodes get larger in WSN. This shows clearly the better scalability of CBGA-based path computing and optimizing scheme employed in this robot routing problem. In particular, comparing with RaceTrack, CBGA can reduce at least 60% of path length for medium- to large-size network. And the reduction is even higher for larger-size networks. The reason is that the final path obtained by RaceTrack is reduced from an initial non-overlapping TSP route, while CBPGA starts with a comparatively shorter overlapping TSP route provided by CA+RS. As the number of sensor nodes increases in a given field, the reduction rate is higher since RaceTrack route should traverse more sensor nodes. On the contrary, much less waypoints exist in the eventual route obtained by CBPGA, since more nodes can be clustered into a cluster and thus more nodes can be accessed from one waypoint.



(a) (b)

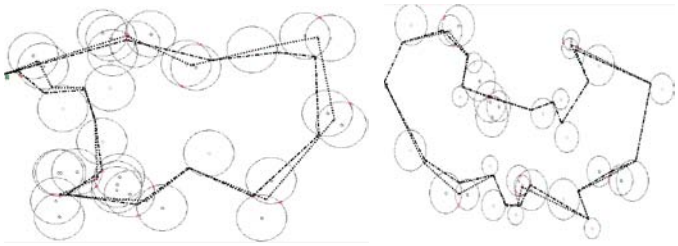
Fig. 6 Length reduction rate of each method compared with CBGA for different numbers of sensor nodes: (a) random and (b) fixed sensing range

The reduction rate in the number of visited nodes as compared with TSP route (number of sensor nodes) is shown in Fig. 7. The reduction rate in the number of visited nodes is different from that in path length, showing that the waypoint selection and the order to visit them are important to obtaining a shorter data-gathering route, in addition to the number of waypoints on the route. Fig. 8 shows an example of routes generated by CBPGA with or without closest waypoint selection with the same order of viewpoint visits. The number of sensors is 30, the population size is 300, probability of crossover = 1, and probability of mutation = 0.2. The sensing radius of all sensors is fixed at 50. Clustering is more advantageous in a dense network, or the radius of sensing range would get larger. The main reason is that the number of visited nodes can be decreased greatly by clustering and each cluster is likely to contain more sensor nodes when there are more overalpping areas of sensing ranges in WSN. Furthermore, the length of chromosome (the number of genes) is reduced, thus improving the efficiency of generating population and computing fitness value in evolutionary process of GA, as will be shown in the following Part 2 simulations.



(a) (b)

Fig. 7 Reduction rate in number of visited nodes for each method as compared with TSP: (a) random and (b) fixed sensing radius



(a) (b)

Fig. 8 The example of paths generated by clustering-based genetic algorithm with (— · — · —) or without (···) closest waypoint selection, respectively, with the same order to visit the waypoints: (a) the sensing radius is 50 and (b) variable sensing radius. (The number of sensors is 30, the population size is 300, probability of crossover = 1, probability of mutation = 0.2.)

B. Part 2: Effect of Parallelism

While the above section shows the scalability and route shortening effect of CBGA, this subsection compares the advantages of exploiting more parallelism, i.e. CBPGA (CBGA equipped with structure of parallel genetic algorithm) and PGA, in addition to the implicit parallelism in GA. In this part of simulations, the sensing range radius is set to 50, and the parameters of the genetic operators are set as the same as in Part 1. First, the effect with various migration intervals is investigated. Fig. 9 is a summary of the simulations related to the effects when migration interval = 1, 100, 200 and 300 respectively. It is found that CBPGA with migration provides a higher convergence rate. Moreover, regarding the effect of the migration interval in Fig. 9, when the migration interval is too short (migration interval = 1, very frequent), CBPGA with migration does not have a higher convergence rate. The same applies when the migration interval is too long (migration interval = 300, rare). When the migration interval is too short, it is easy for the diversity between chromosomes within a slave to decrease. On other hand, when the migration interval is too long, there are relatively fewer chances of getting better chromosomes from other slaves to replace the worst chromosome, and the migration does not make a difference in the effect. Thus, CBPGA with migration is most appropriate when the migration interval is 100 or 200. These results imply that an appropriate configuration of the migration interval is very important; an inappropriate configuration of the migration interval harms the acceleration of convergence. Furthermore, the number of slaves affects the final route discovered by CBPGA, since in general, the number of slaves is related to the number of local optima but at the expense of computing time.

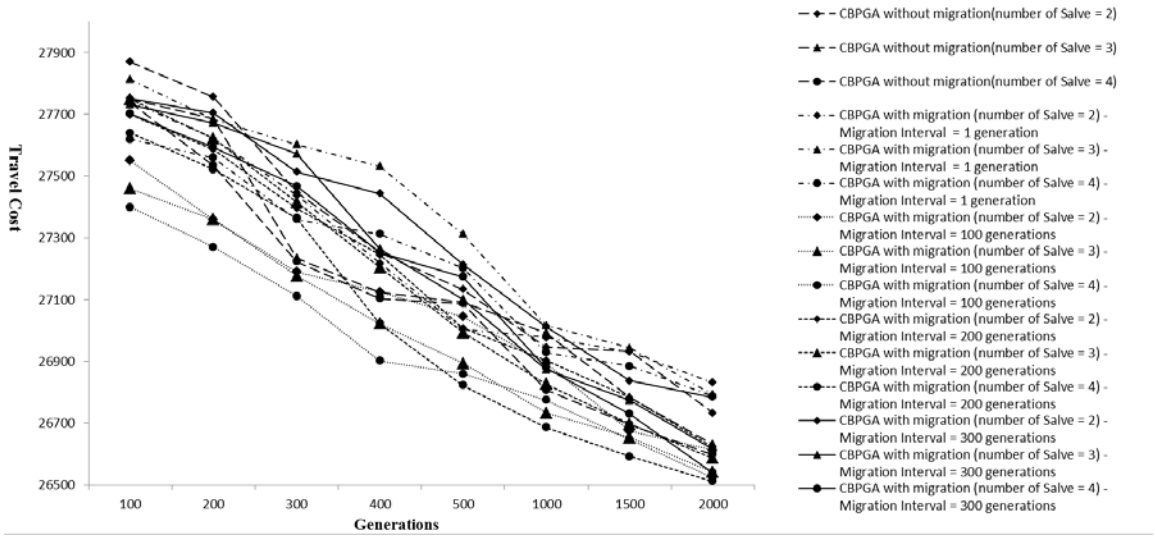


Fig. 9 Convergence rates of CBPGA with and without migration for different numbers of slaves: 2, 3 and 4 when migration interval = 1, 100, 200 and 300, respectively. (The number of sensors is 500, the radius of sensing range is 50, the population size is 1000, crossover probability = 0.2, mutation probability = 0.2, migration probability = 0.2, number of generations = 2000, and the map size is 2000*2000.)

The simulations of CBPGA shown in Fig. 10 (with migration intervals of 100 and 200 generations, 4 slaves) reveal that incorporating migration into the evolution has little influence on the quality of the generated paths in the presented robot routing problem, but is indeed helpful to further acceleration of the convergence. Regarding computation time, it is noted that the length of chromosome of CBPGA could be much shorter than that of PGA when the distribution of sensors is denser (as Part 1 shows), thus the efficiency of generating population and computing time-consuming fitness value is raised in CBPGA. CBPGA can find a better near-optimal route within fewer generations than PGA.

Summarizing the observations in Part 2 of the simulation studies, it can be inferred that the parallel structure provides an improvement in convergence rate of the CBPGA. Therefore, CBPGA is a practical scheme for robot routing in large-scale networks.

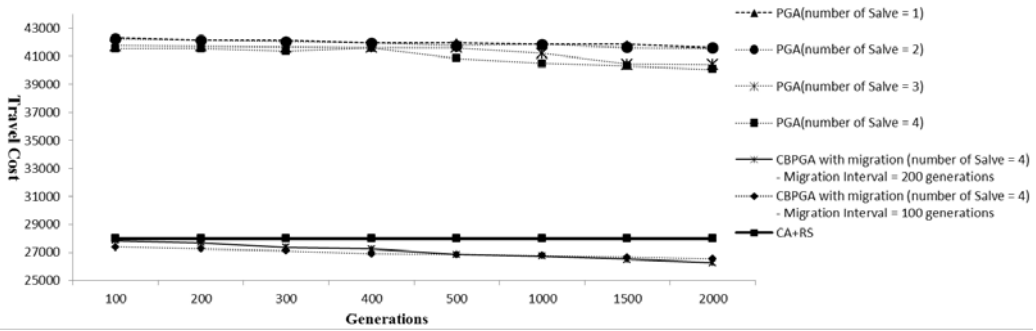


Fig. 10 Convergence of PGA, CBPGA without migration and CBPGA with migration, respectively, with slave number of 4. (The number of sensors is 500, the radius of sensing range of sensors is 50, the population size is 1000, crossover probability = 0.2, mutation probability = 0.2, migration probability = 0.2, generations = 2000, and migration interval = 100, with 200 generations and map size of 2000*2000.

C. Summary of Simulations

The simulations qualitatively reveal that CB(P)GA has better scalability while the network gets larger. Clustering the whole network is helpful in reducing the overall complexity and then GA returns a shorter route. The role of GA is to search for a suboptimal reordering of visited nodes provided by clustering and no matter how the radii of sensing ranges of all sensors vary, GA still works. It is observed that waypoints of the final route are important in reducing the route length further. Moreover, parallelizing the whole process with appropriate migration scheme could accelerate the convergence of the evolutionary process.

D. A Remark on Spanning Tree Covering Algorithm

[4] presented a single-hop data-gathering scheme based on spanning tree covering algorithm (STCA) that ensures every sensor node is in the neighbor set of at least one waypoint on the route, and every sensor can communicate with the mobile robot directly without relay. The reported performance of this scheme is similar to the proposed CA: the higher deployment density and the larger sensing radii, the shorter the route. The waypoints of the final routes could be different in the two schemes. By setting the covering sets to be the sets generated from the clusters derived from CA, Table 2 shows the statistics of the data-gathering schemes of CA and STCA. It can be seen that with this setting, CA outperforms STCA in route length and number of visited nodes, since the number of covering sets is larger than the number of clusters. Fig. 11 depicts one situation where increased number of sets frequently happened in a dense, connected WSN. Fig. 12 shows an example of routes generated by CA and STCA.

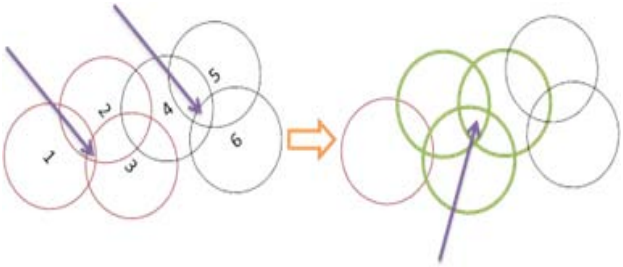


Fig. 11 Generating the covering sets from the clusters produced by CA. The overlapping sets (set 2, 3, 4) of two disjoint clusters (left, thus two waypoints) are grouped into a new covering set (right, thus one additional waypoint).

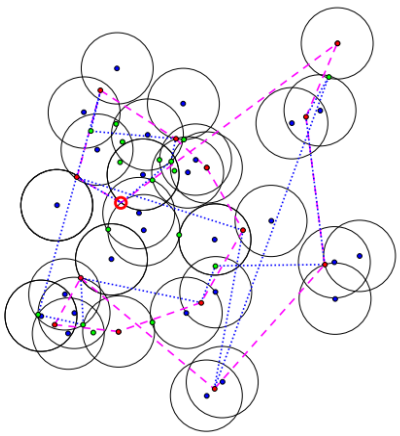


Fig. 12 The routes generated by CA (red dashed line) and STCA (blue dashed line).

TABLE 2 THE ROUTE LENGTH AND REDUCTION RATE IN NUMBER OF NODES FOR CA AND STCA FOR DIFFERENT NUMBERS OF NODES WITH FIXED SENSING RADIUS OF 50. MAP SIZE=2000*2000. (AVERAGE OVER 50 RUNS).

Performance / Node number	CA reduction rate of number of nodes (%)	CA Cost	STCA reduction rate of number of nodes (%)	STCA Cost
100	25.8	18503.9	25.46	21466.6
150	33.2533	21602.1	32.64	25710.6
200	38.65	23701.8	37.5	28644.9
250	42.8	26238.6	41.336	31653.4
300	46.9933	27302.1	45.25333	33474.1
350	49.92	28691.5	47.93714	35678.7
400	52.63	29853.8	50.125	37185.8
450	55.0578	31131.2	52.52	38973.1
500	57.056	31677.5	54.132	40622.4

VI. CONCLUSIONS AND FUTURE WORK

Exploiting a mobile robot for data collection and routing in WSN has garnered recent interests for minimizing both computation and communication costs. Effective use of a data mule to collect data in a wireless sensor network (WSN) involves a problem in robot routing: generating a path along which the mobile robot can retrieve data from sensors in a field while minimizing the overall travel costs (related to latency of data delivery). This robot routing problem is formulated as a Traveling Salesman Problem with Neighborhoods (TSPN). TSPN route design is different from TSP route design in which the relative ordering is the only design factor. The number and locations of waypoints are also important factors in TSPN route design to optimize the distance that the data mule has to travel, in addition to the order of visits. Considering TSPN may yield a larger search space, in this study, all these contributing factors were dealt with and a clustering-based genetic algorithm was proposed and implemented using a parallel architecture of master-slave with migration (CBPGA). Extensive simulations were performed to demonstrate the effectiveness of CBPGA-based data gathering scheme. Based on the simulation results, a clustering-based genetic algorithm with closest-waypoint algorithm can account for simultaneously the number of waypoints, the order to visiting them and their locations; all are contributing factors to path length of data-gathering tour of a data mule. Clustering reduces the search space, GA further shortens the route, and parallelism and migration both accelerate convergence while maintaining diversity. Overall, the combined effects of clustering, parallelism, and migration of CBPGA produce a shorter route with better scalability and consistently that outperform alternative solutions in robot routing problem of WSN. For future work, devising of alternative implementations of GA will be focused on to enhance the performance. Furthermore, it is being planned to implement the CBPGA on a real testbed and analyze the real-world phenomena of wireless devices such as radio irregularity, the changing connectivity and topology of WSN. In general, survival time (energy depletion of sensors) of sensing devices and navigation errors of mobile robot will be deliberated in a more in-depth manner to verify the feasibility and evaluate the performance while the data gathering proceeds in actual sensor network. On the other hand, issues such as collaboration between multiple robots during data collection are left for future study.

ACKNOWLEDGEMENT

This work was supported by National Science Council of ROC under contract NSC 101-2221-E-001-001.

REFERENCES

[1] O. Tekdas, J. H. Lim, A. Terzis and V. Isler. “Using mobile robots to harvest data from sensor fields,” *IEEE Wireless Communications, Special Issue on Wireless Communications in Networked Robotics*, vol. 16(1), pp. 22-28, 2009.

[2] D. Bhadauria and V. Isler, “Data gathering tours for mobile mobots,” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3868-3873, 2009.

[3] A. Jarry, P. Leone, S. Nokoletseas and J. Rolim, “Optimal data gathering paths and energy Balance mechanisms in wireless networks,” *LNCS* 6131, pp. 288-305, 2010.

[4] Ming Ma, Yuanyuan Yang and Miao Zhao, “Tour planning for mobile data-gathering mechanisms in wireless sensor networks,” *IEEE Transactions on Vehicular Technology*, vol. 62(4), pp. 1472-1483, 2013.

[5] Yuldi Tirta, Bennett Lau, Nipoon Malhotra, Saurabh Bagchi, Zhiyuan Li and Yung-Hsiang Lu, "Controlled mobility for efficient data gathering in sensor networks with passively mobile nodes," Section 3.2, pp. 92-113, in *Sensor Network Operations*, Editors: Shashi Phoha, Thomas La Porta, and Christopher Griffin. Wiley-IEEE Press, 2006.

[6] M. Di Francesco, S. K. Das and G. Anastasi, "Data collection in wireless sensor network with mobile elements: a survey," *ACM Transactions on Sensor Networks*, 2011.

[7] M. B. Sares, M. F. M. Campos, D. A. Dutra, V. C. da S. Campos and G. A. S. Pereira, "Hybrid mobile robot navigational strategy for efficient data collection in sparsely deployed sensor network," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2833-2838, 2007.

[8] S. Shue and J. M. Conrad, "A survey of robotic applications in wireless sensor networks," *IEEE Southeastcon*, pp. 1-5, 2013.

[9] Ryo Sugihara and Rajesh K. Gupta, "Path planning of data mules in sensor networks," *ACM Transactions on Sensor Networks*, vol. 8(1), 2011.

[10] Liang He, Z. Chen and J. D. Xu, "Optimizing data collection path in sensor networks with mobile elements," *International Journal of Automation and Computing*, vol. 8(1), pp. 69-77,2011.

[11] T. C. Chen, T. S. Chen and P. Y. Wu, "On data collection in wireless sensor networks," *IEEE Transactions on Systems, Man, an Cybernetics, Part B*, vol. 11(6), pp. 1213-1224, 2011.

[12] B. Yuan, M. Orlowska and S. Sadiq, "On the optimal robot Routing problem in wireless sensor networks," *IEEE Trans. on Knowledge and Data Engineering*, vol. 19(9), pp. 1252-1261, 2007.

[13] E. M. Arkin and R. Hassin, "Approximation algorithms for the geometric covering salesman problem," *Discrete Applied Math*., vol. 55(3), pp. 197-218, 1994.

[14] C. H. Papadimitriou, "The euclidean traveling salesman problem is NP-complete," *Theoretical Computer Science*, vol. 4(3), pp. 237-244, 1977.

[15] E. Aarts and J. K. Lenstra, *Local Search in Combinatorial Optimization*, Wiley Series in Discrete Mathematics & Optimization, 1997.

[16] K. Elbassioni, A. V. Fishkin, Nabil H. Mustafa and René Sitters, "Approximation algorithms for euclidean group TSP," *LNCS* 3580, pp. 1115-1126, 2005.

[17] Wook Ahn Chang and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE Transactions on Evolutionary Computation*, vol. 6(6), pp. 1-7, 1999.

[18] R. V. Kulkarni, A. Förster and G. K. Venayagamoorthy, "Computational intelligence in wireless sensor networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 13(1), pp. 68-96, 2011.

[19] E. Cant´u-Paz, "Efficient and accurate parallel genetic algorithms," Kluwer Academic Publishers, 2001.

[20] S. Yussof, R. A. Razali and O. H. See, "An investigation of using parallel genetic algorithm for solving the shortest path routing problem," Journal of Computer Science, vol. 7, pp. 206-215, 2011.

[21] M. Golub and D. Jakobovic, "A new model of global parallel genetic algorithm," *Information Technology Interfaces (ITI)*," pp. 363-368, 2000.

[22] M. Munetomo, Y. Takai and Y. Sato, "A migration scheme for the genetic adaptive routing algorithm," *IEEE International Conference on Systems, Man, and Cybernetics (ICSMC)*, pp. 2774-2779, 1998.

[23] B. Plamenka and L. Milena, "Migration policies for island genetic models on multicomputer platform," *International Symposium on Algorithms and Computation (IDAACS)*, pp. 143-148, 2007.