

Robot Routing Using Clustering-Based Parallel Genetic Algorithm with Migration

Ko-Ming Chiu

Institute of Information Science
Academia Sinica
Taipei, R.O.C.
chiukoming@iis.sinica.edu.tw

Jing-Sin Liu

Institute of Information Science
Academia Sinica
Taipei, R.O.C.
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. This paper focuses on routing problems with data gathering by a mobile robot in a WSN, also referred to as a Traveling Salesman Problem with Neighborhoods (TSPN) or NP-hard problem. In this paper, we propose a clustering-based parallel genetic algorithm with migration (CBPGA), so that the mobile robot can gather all data from all sensors and the travel costs of the mobile robot clearly decrease. First, a clustering algorithm is used to effectively reduce the number of visited nodes, especially in situations with dense sensor distributions or large sensing radii. Next, the set of visited nodes is encoded as chromosomes by a chromosome generation algorithm (CGA), and the master-slave parallel genetic algorithm with migration is performed to more efficiently generate the near-optimal route. Lastly, a travel cost-reduction scheme is used to remove redundant travel costs. Simulation results confirm that the clustering-based parallel genetic algorithm with migration more efficiently generates a near-optimal route that reduces the travel costs of a mobile robot in robot routing problems with WSN.

Keywords - *Clustering-Based Parallel Genetic Algorithm with migration (CBPGA); Wireless Sensor Network (WSN); Robot Routing Problem; Traveling Salesman Problem with Neighborhoods (TSPN).*

I. INTRODUCTION

Rapid advances in the technology of wireless sensor networks (WSN) have led to its incorporation in a variety of applications, such as environment monitoring, surveillance systems and unmanned space or planet exploration. In unmanned planet exploration, sensors play an important role, and many sensors are often distributed over a geographic area with the capacity to communicate with each other. As a result, numerous studies focus on issues concerning data gathering within these sensor networks [17]. Natural energy restrictions prevent a sensor's communication range from increasing limitlessly until all sensors or an entire area is covered. Here, one possible solution for this data gathering problem is to employ one or more mobile robots that can autonomously gather data from all sensors.

This concept of mobility for data collection and routing has garnered recent interest. In previous research, the main issue has been delivery between cluster heads, which have the disadvantage of consuming more energy during data delivery. These cluster heads deplete energy quickly, and much research has focused on addressing this problem and decreasing the load of the cluster head (LEACH [20], LEACH-C [21], TEEN [22], APTEEN [23]). In general these methods focus on data latency and the data aggregation rate of clusters; however they are unable to guarantee successful data delivery to base stations in environments with sparse sensor distribution. In response to this problem, Yuan et al. [19] presented methods to help a mobile robot collect this data with minimal travel costs. However, their travel cost solution fails to consider situations involving sensors with overlapping communication ranges. Building on this existing research, we propose a clustering-based parallel genetic algorithm with migration (CBPGA) for finding a shorter route for the data-gathering mobile robot. This algorithm allows the robot to gather all data from all sensors while clearly reducing travel costs and finding the near-optimal route within fewer generations. This cost reduction is achieved by decreasing the overall necessary number of visited nodes.

The remainder of this paper is organized as follows. After examining related research 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 simulation results and related analyses. Finally, section VI concludes.

II. RELATED WORKS

The robot routing problem [1], [2], 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) [3], a variant of the Traveling Salesman Problem (TSP) known as the NP-hard problem [5], [7]. First studied by Arkin and Hassin [3], TSPN solutions examine a collection of several regions in a plane, called neighborhoods, and find the shortest tour that visits all neighborhoods. Yuan et al. formulated the problem of collecting sensor data using a robot as a special case of the Traveling Salesperson Problem with Neighborhoods (TSPN), where each neighborhood is a disk region whose center corresponds to a node and the radius is determined by the

communication range [19]. Potential methods, such as the greedy method, the approximation algorithm [4] or Genetic Algorithms (GAs) [6] fail to effectively decrease travel costs for a WSN containing sensors with overlapping communication ranges. The mobile robot need only visit a specific location within the effective communication range of a sensor to download data, and thus does not need to visit the sensor's precise location.

III. MODEL AND PROBLEM DEFINITION

A. Model

The model is formally specified by a set of n sensors (nodes), which are denoted by $S = \{s_1, s_2, s_3, \dots, s_n\}$ and are randomly distributed over a two-dimensional plane. We assume that each sensor s_i can sense a disk region centered at s_i with radius r_s . Each sensor is equipped with omnidirectional antennae to communicate with other sensors or the mobile robot within a disk region centered at s_i with radius r_t . Similarly, the mobile robot (R) can download data from or communicate with other sensors when the mobile robot is inside the sensor transmission range. In this paper, the sensing range (r_s) and transmission range (r_t) of each sensor are the same, and the sensing range of each sensor is identical.

B. Problem

As stated above, the robot routing problem involves a mobile robot moving from a given start position (S) to download data from each sensor with communication range (r_s) in a wireless sensor network. The mobile robot returns to the start position after finishing the data-gathering task. The main objective [2] is to generate a path to guide the mobile robot to download all data from all sensors while minimizing travel costs.

In order to decrease travel costs, a core focus of our study is reducing the number of visited nodes while still ensuring the robot can download all data from all sensors. Equally important is providing the robot the shortest path between nodes. These two concerns represent the heart of our research, and we propose a clustering-based parallel genetic algorithm with migration (CBPGA) as our joint solution.

IV. ADAPTIVE CLUSTERING-BASED PARALLEL GENETIC ALGORITHM WITH MIGRATION (CBPGA)

A. Parallel Genetic Algorithms (PGAs)

Genetic Algorithms (GAs) [6] are a heuristic random search method widely used to find proper solutions to a variety of NP problems within a reasonable amount of time (e.g. Traveling Salesman Problem, Schedule Task and so on). However, when they are applied to more complicated problems (e.g. the number of cities increases in TSP), the time required to find an adequate solution increases. Therefore, past scholars have looked for multiple ways by which to accelerate the convergence rate of GAs. In these ways, the parallel architecture [18], called parallel genetic algorithms (PGAs), is commonly used to improve the convergence rate of GAs. There are three major types of PGAs: (1) the master-slave model, (2) island model and (3) fine-grained model. In the conventional master-slave model, one computing node is designated as the master while the others are designated as

slaves. The master node holds the population and performs most of the GA operations. The master will assign one or more evolutionary tasks (i.e. fitness evaluation, crossover and mutation) to the slaves. This is done by sending one or more chromosomes to the slaves, after which the master will return the best result received from the slave nodes. In a coarse-grained model, the population is divided into several nodes. Each node then has a sub-population on which it executes GA operations. To ensure that successful solutions can be spread to other nodes, the nodes will occasionally exchange chromosomes with each other, called migration.

In a fine-grained model, each node only has a single chromosome, and each node can only communicate with several neighboring nodes. In this case, the population is the collection of all the chromosomes in each node. To execute a genetic operation, a computing node must interact with its neighbors. Since the neighborhoods overlap, eventually the good characteristics of a superior individual can be distributed to the entire population. Fine-grained models have a large communication overhead due to the high frequency of interactions between neighboring nodes. More related details of PGA are introduced in [9]-[11], [18]. In this paper, we adopt the structure of the master-slave model to achieve the objective of parallel processing.

B. Migration Strategy

Large amounts of research on the convergence rate of the PGAs focuses on how to configure appropriate migration parameters. There are five migration parameters: (1) migration frequency, (2) migration rate, (3) information selection, (4) acceptance policy and (5) migration topology [9]. According to observations of these parameters [9], [11]-[13], [18], an appropriate migration configuration can prevent a suboptimal solution from all populations. Thus, a correct migration configuration can help obtain both better and global results with fewer generations. In this paper, we use probabilistic migration to exchange information between slaves and evaluate the effects of migration.

C. Overview of CBPGA

Since TSPN is a NP-hard problem [5], [7], and in order to accelerate the convergence rate of generating the near-optimal shortest route, we combine the structure of the master-slave PGA (parallel genetic algorithm) [9]-[11] with the clustering algorithm, travel cost reduction scheme (CA and RS, explained below), and migration mechanism [10]-[16] that form CBPGA to obtain the near-optimal solution of TSPN. The structure of CBPGA is depicted in Fig. 1. The main tasks of the CBPGA master node include executing the clustering algorithm, creating the initial population by CGA, calculating the fitness value for each chromosome and ensuring each slave shares the best chromosome information with each other.

When a slave receives the population information from the master, the slave performs the whole evolutionary process (i.e. selection operation, crossover operation, mutation operation and migration) until the terminal condition is satisfied. In each slave, the migration operation is performed periodically (e.g. migration intervals = 100 generations and migration probability = 0.2). A slave only returns back with the best result when the evolutionary process is terminated. When the master receives the results from all slaves, it selects the best result from these

received results as the output result. As a final step, a reduction scheme is used to remove redundant travel costs. These processes are introduced in the following.

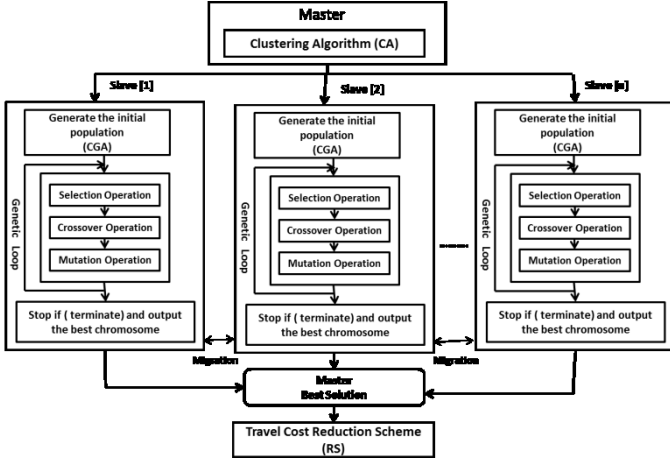


Fig. 1. The structure of the Adaptive Clustering-Based Parallel Genetic Algorithm with Migration (CBPGA)

D. Clustering Algorithm (CA)

Decreasing the number of visited nodes is the fundamental method through which to decrease costs, and this is achieved through the clustering algorithm (CA). Here, the CA classifies several sensors that share a common intersection area as the same cluster. The robot only needs to visit the intersection areas of these clusters to download data from all sensors.

In introducing CA, two tables are created (shown jointly as Table I) to record the relationships between nodes and clusters for quick reference with the clustering algorithm. 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. The details of CA are presented as follows.

TABLE I. 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

Clustering algorithm: the clustering algorithm is used to divide all nodes into a minimal number of clusters. Sensors belonging to the same cluster have a common intersection area. Input: a set of sensors $\mathcal{S} = \{s_1, s_2, s_3, \dots, s_n\}$ and sensing radius (communication range) of all sensors = r_s . Output: a set of clusters $\mathcal{C}_{set} = \{c_1, c_2, c_3, \dots, c_k\}$.

- Initially, $\mathcal{C}_{set} = \emptyset$;
- For each pair (s_i, s_j) , $i, j \in \{1, 2, \dots, n\}$ and $i \neq j$. If $\text{EuclideanDistance}(s_i, s_j) \leq 2*r_s$, go to step 3, otherwise, directly go to step 11.
- If Cluster ID of $s_i \neq$ Cluster ID of s_j , go to step 4, otherwise, go to step 10.
- If Cluster ID of $s_i \neq \text{null}$, go to step 5, otherwise, go to

step 8.

- If Cluster ID of $s_j \neq \text{null}$, go to step 6, otherwise, go to step 7.
- If the sensors, that cluster ID $\in$ Cluster IDs of s_i and s_j (include s_i and s_j), have the same intersection area and the Euclidean distance between any two sensors $\leq 2*r_s$, perform $\text{MergeToNewCluster}(s_i, s_j)$, otherwise, directly go to step 11.
- If the sensors, that Cluster ID = Cluster ID of s_i , and s_j have the same intersection area and the Euclidean distance between any two sensors $\leq 2*r_s$, perform $\text{MergeToNewCluster}(s_i, s_j)$ and go to step 11, otherwise, perform $\text{AddNewCluster}(s_j)$ and go to step 11.
- If Cluster ID of $s_j \neq \text{null}$, go to step 9, otherwise, directly go to step 11.
- If the sensors, that Cluster ID = Cluster ID of s_i , and s_j have the same intersection area and the Euclidean distance between any two sensors $\leq 2*r_s$, perform $\text{MergeToNewCluster}(s_i, s_j)$ and go to step 11, otherwise, perform $\text{AddNewCluster}(s_i)$ and go to step 11.
- If Cluster IDs of s_i and $s_j = \text{null}$, perform $\text{MergeToNewCluster}(s_i, s_j)$ and go to step 11, otherwise, directly go to step 11.
- If all sensors are divided into a set of clusters $\mathcal{C}_{set}$, return back with $\mathcal{C}_{set}$, otherwise, go to step 2.

In the Euclidean plane, if $p = (p_1, p_2)$ and $q = (q_1, q_2)$, the function $\text{EuclideanDistance}(p, q)$ in CA is given by

$$\text{EuclideanDistance}(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$$

An illustration of the $\text{MergeToNewCluster}(s_i, s_j)$ function is shown as Fig. 2, and adheres to the following process. First, we obtain the Cluster IDs of s_i and s_j respectively. According to these IDs, we can find all corresponding nodes. Next, we create a new Cluster ID and the Node ID filed within the Cluster-Nodes table is filled with the corresponding nodes obtained previously. We then delete the rows of the Cluster IDs of s_i and s_j in the Cluster-Nodes table. Finally, the Cluster IDs of these corresponding nodes are all replaced with new Cluster IDs.

In addition to $\text{MergeToNewCluster}(s_i, s_j)$, the function $\text{AddNewCluster}(s_i)$ denotes that only s_i is added to a new cluster. This function is used to add a new Cluster ID in the Cluster-Nodes table. The value of the Node IDs field is filled with the Node ID of s_i , and the value of the Cluster ID field is replaced by a new Cluster ID in the Cluster-Nodes table. Through the above clustering algorithm, a set of sensors is divided into a set of clusters $\mathcal{C}_{set}$. The sensors within each cluster have the same intersection area. An illustration of this clustering of sensors is shown in Fig. 3. For example, this figure shows how one sensor does not have a common intersection area with all sensors of Cluster-2. Therefore, this sensor is classified into a new cluster, Cluster-4.

In this paper, we compare two methods for reducing travel costs (the greedy method and the approximation algorithm [4]) with our proposed method. In the greedy method, a mobile robot starts from a start position (S), travels

to the location of the next sensor until all sensors are visited, and then returns to the original start position (S). The main difference between the greedy method and the approximation algorithm [4] is that the mobile robot need not visit the precise location of each sensor to gather data. The approximation algorithm can remove some redundant travel paths of the mobile robot, so as to decrease travel costs. However, neither alternative algorithm reduces the number of visited nodes, as in both cases the mobile robot must still essentially visit each sensor. In contrast, the clustering algorithm proposed in this paper allows the mobile robot to simply visit locations within the intersecting areas of sensor ranges, rather than visiting each node individually. When the clustering of nodes is complete, we can obtain a route generated by the following steps.

1. Take the start position (S) of the mobile robot as the first position.
2. If the number of nodes in the closest cluster > 1 , take the middle node of the closest cluster as the next visited node and go to step 4. Otherwise, go to step 3.
3. If the number of nodes in the closest cluster $= 1$, take the node of the closest cluster as the next visited node and go to step 4.
4. If all clusters have been visited, go to step 4. Otherwise, go to step 2.
5. Take the start position (S) of the mobile robot as the last position.

The route generated by the above processes is broadcast by the master to the slaves as a feasible route.

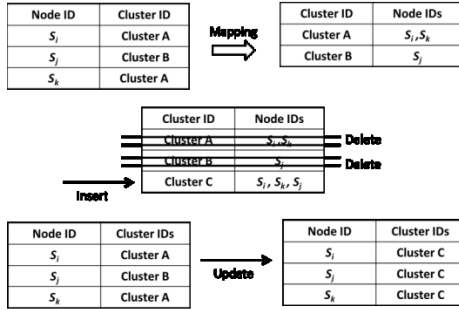


Fig. 2. The illustration of MergeToNewCluster(S_i, S_j) function.

E. Parallel Genetic Algorithm

Details of our current implementation of PGA for finding a shorter route of a mobile robot in WSN are introduced as follows.

1). Genetic Representation

Each chromosome represents a feasible route, and is encoded as an ordered sequence of visited nodes. Because a mobile robot will return to its start position to transmit data to the base station after completing its data-gathering tasks, the start position and the end position are the same. In other words, the first gene and the last gene of each chromosome in a population are fixed, and are always the start position. Moreover, there is a constraint for each chromosome. In each chromosome, each visited node is allowed to appear once, except for the first and last genes. If a chromosome contains

the repeated visited nodes, this chromosome is regarded as an invalid chromosome.

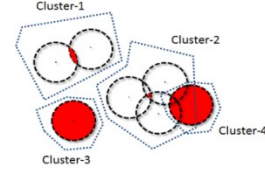


Fig. 3. This figure shows four clusters.

2). Initial population

A population is composed of chromosomes. Here, the generation of an initial population with high diversity between chromosomes is an important issue; an initial population with high diversity has a higher chance of evolving great child chromosomes, and each population chromosome generated serves as a feasible solution. In order to generate populations with high diversity, this paper proposes a Chromosome Generation Algorithm (CGA) based on a permutation tree to generate non-repeated chromosomes. The algorithm is as follows.

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

Input: A feasible route (P) and population size (P_{Len}).

Output: the non-repeated chromosomes in descending order of route length.

//Initial Configuration- Start

$P = S [S_1, S_2, S_3, \dots, S_n] S;$

$N_e = \{S_1, S_2, S_3, \dots, S_n\};$

$N_{Len} = \text{Length of } \{S_1, S_2, S_3, \dots, S_n\};$

if ($P_{Len} > N_{Len}!$)

{
 $P_{Len} = N_{Len}!;$
}

We can obtain a number (N) and $\min([N!]) \geq P_{Len};$

Randomly choose a position set $P_s = \{p_1, p_2, p_3, \dots, p_N\};$

Then, Get the corresponding nodes $E_s = \{E_1, E_2, E_3, \dots, E_N\}$ of $N_e;$

//Initial Configuration- End

//Variables Declaration - Start

ListOfPermutation = $\emptyset;$

Permutation = new int[N];

level = -1;

//Variables Declaration - End

//Function Declaration - Start

GeneratingPermutationTree(k)

{

$level++;$

Permutation[k] = **level**;

if ($level == N$)

 {

 int[] **newPermutation** = new int[N];

newPermutation $\leftarrow$ **Permutation**;

if (**newPermutation** not exists in **ListOfPermutation**)

 {

ListOfPermutation.Add(**newPermutation**);

 }

 } **else**

 {

for (int $i = 0; i < length; i++$)

 }

```

{
    if (Permutation[i] == 0)
    {
        GeneratingPermutationTree(i);
    }
}
level -= 1;
Permutation[k] = 0;
}
//Function Declaration - End
//Pseudo-Code - Start
GeneratingPermutationTree(0);
NewPopulation = GeneratingNewPopulationAndCalEuclideanDistance
(S, Ps, Ne, PLen);
SortByEuclideanDistanceInDescendingOrder(NewPopulation);
If(P not exist in NewPopulation)
{
    if (P is superior to the last chromosome in NewPopulation )
    {
        NewPopulation.Add(NewPopulation);
        Remove the last chromosome in NewPopulation;
        SortByEuclideanDistanceInDescendingOrder(NewPopulation);
    }
}
Return NewPopulation;
// Pseudo-Code - End

```

The above processes can generate non-repeated chromosomes, guaranteeing the desired diversity between initial chromosomes. Because the solution combinations are sorted with the Euclidean distance in descending order, we can choose initial chromosomes randomly from the non-repeated chromosomes generated by the CGA. If the size of the population needed is larger than the total number of non-repeated chromosomes, the population size is set to the number of non-repeated chromosomes that will guarantee that these chosen solution combinations (chromosomes) are not identical. When the population size is significantly less than the number of non-repeated chromosomes, it is a waste to generate all non-repeated chromosomes. In order to decrease the overhead of generating all non-repeated chromosomes, we first select a number (N) so that the generated non-repeated chromosomes is larger than or equal to the population size (i.e. $\min([N!]) \geq$ the population size). Then, the N visited nodes are randomly selected from the visited nodes set, and their positions are recorded in the visited nodes set. A detailed illustration is shown in Fig. 4. Through these processes, we can obtain the required number of solution combinations without generating all non-repeated chromosomes, thus decreasing overhead.

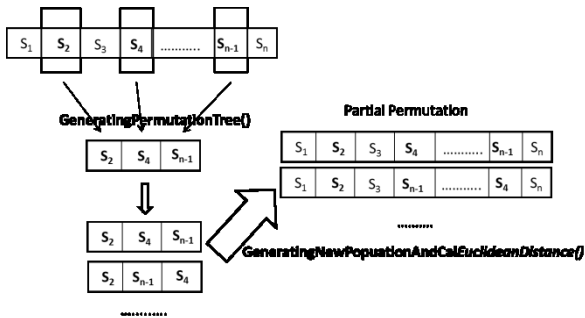


Fig. 4. Illustration of generating a partial permutations from the set of sensor nodes.

3). Fitness Function

Each chromosome in the population is evaluated by a fitness value. This fitness value indicates how strong the solution is for a particular chromosome. The fitness function f_i denotes the fitness value of the i th chromosome, and is designed as follows.

$$f_i = \frac{1}{c_i}$$

Let c_i denotes the total cost of the i th chromosome. The total cost is defined as the sum of the Euclidean distances of a specific chromosome. A higher fitness value is given for a shorter route (better chromosome).

4). Selection operation

In order to generate the next generation of chromosomes, a mating pool that consists of the all 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. 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 choosing chromosomes in the mating pool will be selected as a parent. To ensure the parents are valid (i.e. no repeated visited node), the selection operation is repeated until two valid parents are chosen.

5). Crossover operation

The crossover operation is a probabilistic process that exchanges information between two parent chromosomes for generating two new child chromosomes. The crossover operation is used to explore new solution and hopefully to be able to find better solution in this process. In order to perform the crossover operation, the two parent chromosomes from mating pool will be selected by using 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.) Two new child chromosomes generated are subjected to the constraint of valid chromosome. If the child chromosomes are invalid, the invalid child chromosomes will be discarded and the valid child chromosomes will be reserved. If the new child chromosome is valid, the new chromosome will be placed in appropriate location and the worst chromosome will be discarded. Otherwise, the new child chromosome will be discarded.

6). Mutation operation

The mutation operation is used to create diversity in the population. In this paper, each chromosome produced by the crossover operation has a small chance to be mutated based on a mutated probability. For each chromosome that is chosen to be mutated, two mutation points will be chosen randomly with equal probability. Then, the genes of two mutation points are swapped to form a new mutated chromosome. If the mutated chromosome is valid, the mutated chromosome is placed in the

appropriate location and the worst chromosome is discarded. Otherwise, the mutated chromosome is discarded.

7). Migration operation

In the parallel genetic algorithm, the migration operation is the most common method to provide a mechanism of occasional genetic information exchange between slaves [10]-[16]. By this mechanism, a slave can share their own best chromosome with other slaves in the parallel genetic algorithm. Existing research finds that the results using migration performed significantly better than those without migration. In this paper, first a migration pool is created. 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 from a slave, it will compare it with the original chromosome. If the newly generated chromosome is better than the original, the original chromosome will be replaced by the new chromosome. This new chromosome is then broadcast to a slave chosen randomly. When a slave receives the new chromosome, the new chromosome will be placed in an appropriate location and the worst chromosome will be discarded.

F. Travel Cost Reduction Scheme (RS)

The travel cost reduction scheme (RS) is used to further remove redundant travel costs. In sparse sensor distributions, there are many clusters which only include one sensor, as illustrated by Cluster-3 and Cluster-4 in Fig. 3. In order to further reduce robot travel costs, the position of the closest cross point of a line and a circle is generally the first candidate of a visited node. Commonly, the position of the closest cross point is obtained by solving simultaneous equations composing of a linear equation, $\mathbf{ax} + \mathbf{by} + \mathbf{c} = \mathbf{0}$, and a circle equation $(\mathbf{x} - \mathbf{h})^2 + (\mathbf{y} - \mathbf{k})^2 = \mathbf{r}^2$; the closest cross point is the common solution of the linear and circle equations. In this paper, the RS obtains the closet cross point without solving any linear or circle equations, as illustrated in Fig. 5. It instead uses the triangular relationship to find the closest cross point. That is, given a node P and a circle centered at O with radius r_s , the coordinates of the closest cross point M(x' , y') can be obtained by the RS.

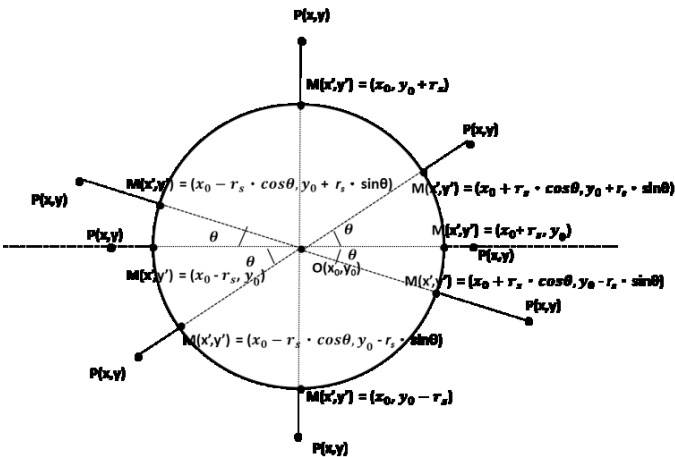


Fig. 5. Illustration of the travel cost reduction scheme (RS)

V. SIMULATION RESULTS

In our simulations, the sensors are randomly distributed over a two-dimensional map with size set to 2000*2000. Here, each value is the average of 50 samples. Simulation experiments include two parts to reevaluate the clustering-based parallel genetic algorithm with migration. The first part involves the effects of the CA and RS. Two previously proposed methods (i.e. the greedy algorithm and the approximation algorithm [4]) are compared with CA and CA plus RS, with identical sensor distributions for the number of sensors, and the radii of the sensing range set to 10, 30 and 50 respectively. In the second part, the proposed clustering-based parallel genetic algorithm with migration (CBPGA) is compared with the parallel genetic algorithm (PGA) with the same parameters, to isolate the effect of migration. The simulation results are summarized in Fig. 6.

A. Effect of CA and CA plus RS

In the simulation, the route generated by CA as mentioned previously. Then, the process for route generation by CA plus RS is similar to CA, except for step 3. The third step of CA plus RS is revised as follows.

3. If the number of nodes of the closest cluster = 1, take the cross point of the closest cluster as the next visited node and go to step 4.

Table II shows that clustering greatly decreases the number of visited nodes when the density of the sensor distribution increases (the sensing range of each sensor is identical). Thus, it is obvious that when using the clustering algorithm, the number of nodes visited by the mobile robot is reduced, and the travel cost or consumption of energy will decrease correspondingly. Moreover, Fig. 6 shows that given identical sensor distributions and sensing ranges (at 10, 30 and 50 respectively), the travel costs of CA plus RS compares favorably relative to CA alone, the greedy algorithm, and the approximation algorithm. Thus, we conclude that CA plus RS is the best method for finding the most efficient route.

TABLE II. THE DIFFERENCE COMPARISON OF VISITED NODES BETWEEN CA AND NON-CA (THE RADIUS RT OF SENSING RANGE OF SENSOR = 50)

No clustering Algorithm	100	150	200	250	300	350	400	450	500
Clustering Algorithm	57	98	119	142	166	201	238	256	271
Average Reduction Rate of Visited Node (%)	43	38	40.5	43.2	44.67	42.57	40.5	43.11	45.8

B. Evaluation of CBPGA

While the above section compares CA plus RS with CA alone, this section compares the travel cost reductions of the CBPGA (CA plus RS equipped with structure of parallel genetic algorithm) with CA plus RS, and PGA. In our simulations, the radius (r_s) of the sensing range of a sensor is set to 50, and the parameters of the genetic operators are set as follows: the number of sensors = 500, crossover probability = mutation probability = 0.2, migration probability = 0.2 and the population size is set to 1000. First, the effect with various migration intervals (i.e. migration intervals = 1, 100, 200 and

300 respectively.) is investigated. Fig. 7 is a summary of our simulations related to the effect with various migration intervals. From Fig. 7, we find that CBPGA with migration provides a more accelerated convergence rate. Moreover, regarding the effect of the migration interval in Fig. 7, when the migration interval is too short (migration interval = 1, very frequent), CBPGA with migration does not have a more accelerated convergence rate. The same applies when the migration interval is too long (migration interval = 300, rare). Thus, CBPGA with migration is most appropriate when the migration interval = 100 or 200.

When the migration interval is too short, it is easy for the diversity between chromosomes within a slave to decrease, so that the better chromosome generated by evolutionary processes is similar to the native chromosome. Thus, it is difficult to generate better chromosomes with high diversity. On other hand, when the migration interval is too long, there are relatively fewer chances of getting better chromosomes from other slaves to replace a worst chromosome, and the effect with migration is similar to without migration.

Next, in our simulations, the radius (r_i) of the sensing range of a sensor is set to 50, and the parameters of the genetic operators are set as follows: the number of sensors = 500, crossover probability = mutation probability = 0.2, migration probability = 0.2, migration intervals = 100 and 200 generations, and the population size is set to 1000. From Fig. 8, we find that with the same number of slaves, the CBPGA identifies a better route within fewer generations than PGA. Moreover, CBPGA with more slaves also finds a near-optimal route within fewer generations than PGA and CA plus RS. Fig. 8 reveals that migration is indeed helpful for accelerating the CBPGA convergence rate. With the dense sensor distribution in Fig. 8, CBPGA with migration and CBPGA without migration are both better than CA plus RS in terms of convergence rate. These results imply that an appropriate configuration of the migration interval is very important; an inappropriate configuration of the migration interval harms convergence rate acceleration. Additionally, incorporating migration into the evolution has little effect on the quality of the generated path in the robot routing problem, but is indeed helpful with further acceleration of the convergence rate. Summarizing the observations above, we can infer that the parallel structure provides an improvement in convergence rate acceleration for the CBPGA.

VI. CONCLUSION

This paper proposes a clustering-based parallel genetic algorithm with migration (CBPGA) for solving the robot routing problem of WSN. Based on the simulation results obtained, clustering reduces the search space, and parallelism and migration both accelerate convergence while still maintaining diversity. Overall, the combined effects of clustering, parallelism, and migration with CBPGA effectively and efficiently find the near-optimal route within fewer generations than other evolutionary methods, such as GA, PGA, and CBPGA without migration. Furthermore, this combination obtains a better route than non-evolutionary

methods (i.e. greedy method, approximation method [4], CA, CA+RS) in robot routing problem of WSN.

REFERENCES

- [1] O. Tekdas, J.H. Lim, A. Terzis, and V. Isler. "Using mobile robots to harvest data from sensor fields," IEEE Wireless Communications, Accepted to the Special Issue on Wireless Communications in Networked Robotics, vol. 16, Issue 1, 2009, pp. 22–28.
- [2] D. Bhadauria and V. Isler, "Data gathering tours for mobile robots", IEEE/RSJ International Conference on Intelligent Robots and Systems 2009, 2009, pp. 3868–3873.
- [3] E.M. Arkin and R.Hassin, "Approximation Algorithms for the Geometric Covering Salesman Problem," Discrete Applied Math., vol. 55, no.3, 1994, pp. 197–218.
- [4] K. Elbassioni, A. V. Fishkin, Nabil H. Mustafa and René Sitters, "Approximation Algorithms for Euclidean Group TSP," LNCS 3580, 2005, pp. 1115–1126.
- [5] C.H. Papadimitriou, "The Euclidean Traveling Salesman Problem Is NP-Complete," Theoretical Computer Science, vol. 4, no. 3, 1977, pp. 237–244.
- [6] Chang Wook Ahn; Ramakrishna, R.S., "A genetic algorithm for shortest path routing problem and the sizing of populations," IEEE Transactions on Evolutionary Computation, vol. 6, issue 6, 1999, pp. 1–7.
- [7] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to Algorithms," MIT Press, Cambridge, Massachusetts, 2001.
- [8] R. V. Kulkarni, A. Förster, and G. K. Venayagamoorthy, "Computational intelligence in wireless sensor networks: A survey," IEEE Communications Surveys & Tutorials, to appear in volume 13, 2011, pp. 1 – 29.
- [9] S. Yussof, R. A. Razali, O. H. See, A. A. Ghapar and M. M. Din, "A Coarse-grained Parallel Genetic Algorithm with Migration for Shortest Path Routing Problem," IEEE International Conference on High Performance Computing and Communications (HPCC) 2009, 2009, pp. 615 – 621.
- [10] Golub M. and Jakobovic D., "A New Model of Global Parallel Genetic Algorithm," Information Technology Interfaces (ITI) 2000, 2000, pp. 363 – 368.
- [11] Laurens Jan Pit, "Parallel Genetic Algorithms," Master's Thesis, Department of Computer Science, Leiden University, 1995.
- [12] Plamenka B. and Milena L., "Migration Policies for Island Genetic Models on Multicomputer Platform," International Symposium on Algorithms and Computation (IDAACS) 2007, 2007, pp. 143 – 148.
- [13] Wen-Yang Lin, Tzung-Pei Hong and Shu-Min Liu, "On Adapting Migration Parameters for Multi-Population Genetic Algorithms", IEEE International Conference on Systems, Man, and Cybernetics (ICSMC) 2004, 2004, pp. 5731 - 5735.
- [14] Munetomo M., Takai Y. and Sato Y., "A Migration Scheme for the Genetic Adaptive Routing Algorithm," IEEE International Conference on Systems, Man, and Cybernetics (ICSMC) 1998, 1998, pp. 2774 – 2779.
- [15] Matsumura T., Nakamura M., Miyazato D., Onaga K. and Okech J., "Effects of Chromosome Migration on a Parallel and Distributed Genetic Algorithm", Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN) 1997, 1997, pp. 357 – 361.
- [16] Rebaudengo M. and Sonza Reorda, M., "An experimental analysis of the effects of migration in parallel genetic algorithms", Euromicro Conference on Parallel, Distributed and Network-Based Processing (EMPDP) 1993, 1993, pp. 232 – 238.
- [17] Kazi C. R., "A Survey on Sensor Network", Journal of Convergence Information Technology (JCIT) 2010, Vol 01, Issue 01, 2010, pp. 76 – 87.
- [18] E. Cant'ú-Paz. "Efficient and accurate parallel genetic algorithms," Kluwer Academic Publishers, 2001.
- [19] B. Yuan, M. Orlowska, and S. Sadiq. "On the optimal robot routing problem in wireless sensor networks", IEEE Trans. on Knowl. and Data Eng., Vol 19, Issue 9, 2007, pp. 1252–1261.

- [20] Wendy Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks", Proceedings of the Hawaii International Conference on System Sciences, 2000, January.
- [21] Endi B. Heinzelman, Anantha P. Chandrakasan, and Hari Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks", IEEE Transactions On Wireless Communications, Vol. 1, No. 4, 2002, October.

- [22] Arati Manjeshwar, Dharma P. Agrawal, "TEEN: A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks," ipdps, p. 30189a, 15th International Parallel and Distributed Processing Symposium (IPDPS'01) Workshops, 2001.
- [23] A. Manjeshwar and D.P. Agrawal, "APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks," in the Proceeding of the 2nd International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, Ft.Lauderdale, FL, 2002, April.

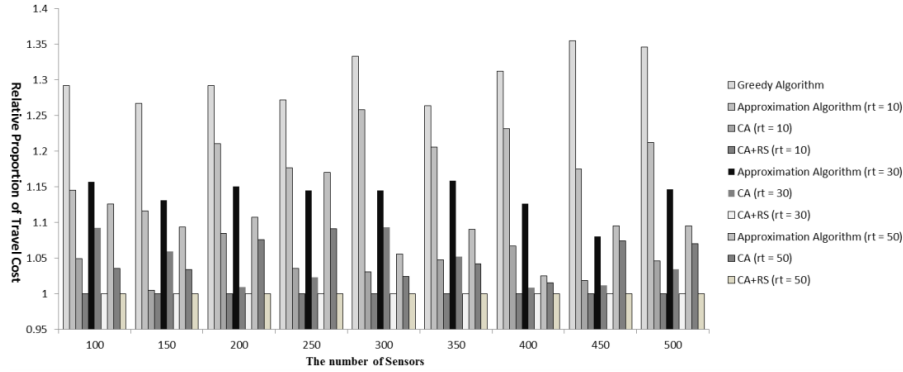


Fig. 6. This shows the relative proportion of travel cost among greedy algorithm, the approximation algorithm [4], CA and CA+RS for different number of sensors. (The different radius (r_i) of sensing range of sensor: 10, 30 and 50)

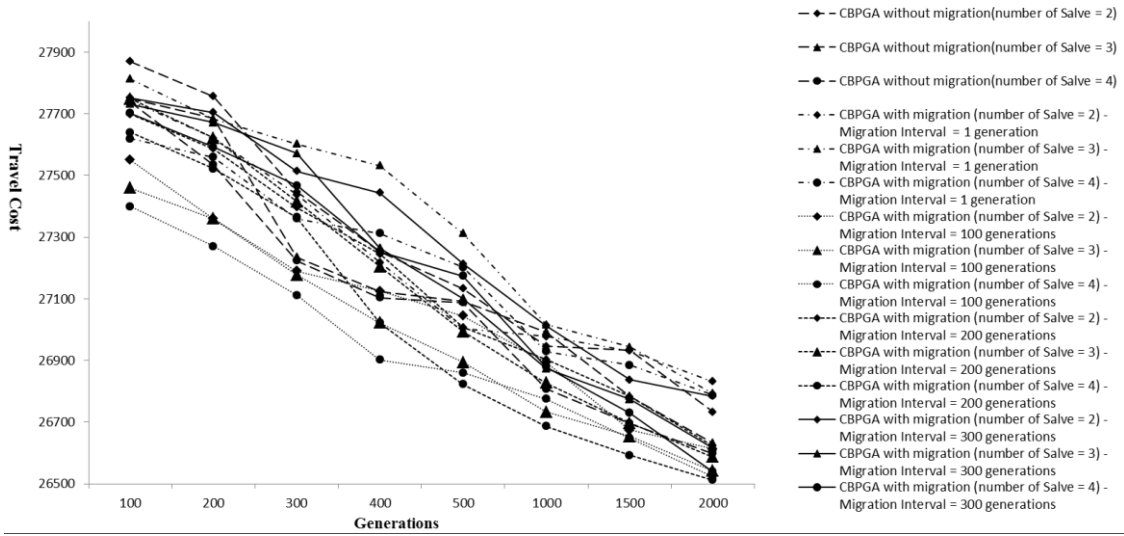


Fig. 7. This shows the difference of convergence rates of CBPGA without migration and CBPGA with migration for different number of slaves: 2, 3 and 4 and migration intervals = 1, 100, 200 and 300 respectively. (The number of sensors = 500, the radius of sensing range of sensor = 50, the population size = 1000, crossover probability = 0.2, mutation probability = 0.2, migration probability = 0.2 and generations = 2000.)

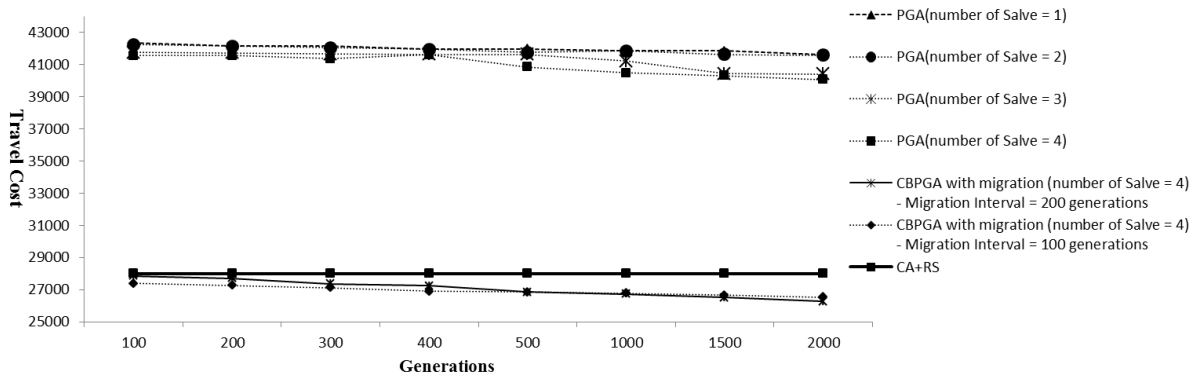


Fig. 8. This shows the difference of convergence rates of CA plus RS, PGA, CBPGA without migration and CBPGA with migration for number of slaves = , 4 respectively. (The number of sensors = 500, the radius of sensing range of sensor = 50, the population size = 1000, crossover probability = 0.2, mutation probability = 0.2, migration probability = 0.2, generations = 2000 and migration intervals = 100, 200 generations)