International Journal of Computational Intelligence Systems, Vol. 7, No. 6 (December 2014), 1109-1122

Optimal Robot Path Planning for Multiple Goals Visiting Based on Tailored Genetic Algorithm

Fei Liu, Shan Liang, Xiaodong Xian

College of Automation, Chongqing University
No. 174, Shazhengjie, Shapingba, Chongqing, China
E-mail: liufei299@yahoo.com;lightsun@cqu.edu.cn;xxd@cqu.edu.cn

Received 13 July 2013

Accepted 13 March 2014

Abstract

In real applications, mobile robot may be commanded to go to multiple goals to execute special commissions. This study analyzes the particular properties of this multiple goals visiting task and proposes a novel tailored genetic algorithm for optimal path planning for this task. In proposed algorithm, objectives for evaluating the path are energy consumption and idle time that are proposed in our previous work. Under the constraint of energy consumption, it will generate an optimal path that comprises as more goals as possible and as less idle time as possible. In this algorithm, customized chromosome representing a path and genetic operators including *Repair*, *Cut* and *Deletion* are developed and implemented. Afterwards, simulations are carried out to verify the effectiveness and applicability. Finally, analysis of simulation results is conducted and future work is addressed.

Keywords: genetic algorithm;mobile robot; optimal path planning; multiple goals visiting; idle time

1. Introduction

Mobile robots have been developed for many real-life tasks such as automatic patrolling in a transformer substation [1], welding automatically in a production line [2] and tour guiding [3,4]. For all the applications, path planning plays an important role in navigating robot to execute missions [5,6]. Further, it is generally occurred that more than one accessible path can be found, thus a strategy is necessary for selecting the optimal or near-optimal path. In recent years, the important issue of optimal path planning has attracted considerable attentions.

1.1. Cases of path planning

Generally, path planning is to find a suitable collision-free path for a robot moving from a start point to a designated goal [7,8]. In this situation, there are just one start location and one goal. However, in different applications, there are four other cases of path planning according to the number of robots, start points and goals:

(i) Path planning for one robot that starts from a point, and chooses a goal from multiple goals to move to. For example, when needing to recharge, the robot should select a charging station from all the stations to go to [1];

(ii) Path planning for one robot that moves from a start point and arrives at a destination while during this course it must visit parts of the specified goals, for example, to pick up loads and carry them to the target location [9]. Specially, if the robot must visit all the nodes in the environment and finally return to the initial

point, it is well known as the routing problem of TSP (Traveling Sales Problem) [10];

(iii) Path planning for Multiple robots that leave from the same start point and go towards the same goal [11];

(iv) Path planning for Multiple robots that start from different initial points and move to different goals [12,13].

The problem that this study will concentrate on is more close to TSP in which there are many goals. Further, the most alike research was described in [9] where there are a start point and a destination and the robot chooses parts of the other nodes to visit. Actually this can be deemed as a part of our problem since in our work there exists several goals and for visiting each goal, it is similar to that starting from a new start point and going to a new goal. However, compared with these works it has five important specific properties:

(i) The ultimate goal is not designated, thus every goal may be the final destination;

(ii) All the problems are usually solved by using a graph-based environment in which goals are represented by nodes. For TSP, all the nodes are goals, while in this study goals are just a part of the nodes;

(iii) No priority is set for goals and therefore the robot can visit the goals in any order;

(iv) The robot may arrive at a goal for more than once (only the first time arriving at a goal is for visiting this goal), which is different from other researches since they require that the robots visit each goal only once strictly;

(v) Since the robot may have not sufficient energy to visit all the goals,the optimal path is allowed to involve not all the goals.

To the best of the authors' knowledge, no research has investigated on this special problem with the five properties listed above.

1.2. Objectives for optimal path determination

The optimal path planning task can be described as an optimization problem in which a single objective or multiple objectives are employed. Among researches about optimal path planning, mainly path length is used for evaluating a path [14,15]. However, when various features of outdoor environment are considered such as friction and gravity, other criteria are proposed for determining an optimal path. For example, Wang et al. intend to plan a time-optimal trajectory for the mobile robot [16]. When finding optimal paths on terrains for a mobile robot, Sun et al. use energy consumed due to friction and gravity as the cost of a path [17]. Liu et al. also treat energy consumption as the central factor in the cost function when developing the global path planner for the mobile robot [18]. Mei et al. [19] consider the energy expended on rotating, since in the environment with many walls and corners, it may cause much energy consumption if rotating frequently. For planning an optimal path to multiple goals, Lobaton et al. took retracing into consideration [20]. In previous research on optimal path planning, on consideration of road attributes including length, road grade, surface roughness and the set of speed hump, we have studied optimal path planning based on energy consumption [21]. Further, considering influence of vibration on mobile robot induced by motion, we proposed the decision factor - idle time (non-working time) as the cost of a path, which is proven to be more comprehensive for evaluating a path [1].

In many cases multiple objectives are considered simultaneously. In terrains, as part of path segments are easy to pass while some are difficult, authors use two factors, namely path length and difficulty for evaluation of paths [22,23]. It may occur that there is no longer a single optimal solution but rather a whole set of possible solutions of equivalent quality if more than one objective is considered. When dealing with multiple objectives, researchers have created effective methods for various situations. A general way is to assign a weight to each objective, and then use the sum or product of the weighted value of each objective as the decision factor for evaluating a path [11,24,25]. The advantage of this method is that it is simple and easy to realize. However in some situations, it is difficult to give an exact weight for each factor, so other methods dealing with the objectives are proposed. Takanori Shi-

bata et al. use a fuzzy set to determine the fitness of a string (representing a path) where each value is decided by effects of two objectives, i.e., time and load [9]. Lexicographic method is another effective approach with which the objective functions are arranged in order of importance [26]. In addition, it is reasonable to pursue an optimal solution according to one criterion while satisfying other objectives that are used as constraints, which is called "bounded objective function method" [26]. Moreover, Gideon Avigad et al. have studied the sequential optimization-constraint multi-objective problems, where different optimal solutions are generated in accordance with different planning demands [27]. In this work, both lexicographic method and bounded objective function method will be employed to deal with the two factors, i.e., energy consumption and idle time proposed in previous studies [1,21].

1.3. GA-based path planning

A majority of researches have concerned on optimal path planning and many conventional techniques including potential field method, visibility graph and Voronoi roadmap are used [28]. Recently, various kinds of artificial intelligence methods like genetic algorithms, neural networks, fuzzy logic method, particle swarm optimization and ant colony optimization have been proposed for optimal path planning [29,30]. In this work, Genetic algorithm (GA) is adopted and tailored to solve concrete problem. Genetic algorithm, based on the mechanism of natural selection and natural genetics, was first developed in the 1970s by Holland [31]. It is an evolutionary optimization method and is proven to perform well in optimal path planning [32]. To use GA, one should first find a pattern to express the feasible solutions, which is called chromosome. Besides, it is necessary to create a fitness function to evaluate each solution. The most challenging part is developing some appropriate genetic operators acting on the population of each generation that is the set of solutions. After evolving for certain generations, the optimal one will be determined by a criterion.

For diverse applications, due to the differentiation of problems, various modifications are made based on basic GA to solve concrete problems. In many occasions, researches use fixed-length chromosome to represent a path [33,34]. While in other circumstances, variable-length chromosomes are adopted. For example, in a grid-based environment, authors use string of cells to describe a path whose length is unfixed [11,35,36,37]. Meanwhile, different forms of fitness functions are created due to the fact that different objectives should be considered in respective application, such as path length [14], energy consumption [18], time consumption [16], smoothness and safety [38]. The key for evolution are the genetic operators. Traditionally, three operators, i.e., *selection*, *crossover* and *mutation* are employed nearly in all applications [36]. They play significant role in adding diversity to the population and therefore are in favour of finding the global optimal solution. Apart from them, customized genetic operators are often established according to different purposes. For example, to make a feasible solution better, operator *improvement* is designed, which will randomly choose a node, and search in neighbouring grids of the node, and move it to a better location [11]. This function is also realized by an operator with the name *repair* [22]. However, there exists an operator called *repair* [11], while it is used for make infeasible path feasible. Besides, *deletion* is employed to eliminate duplicate nodes existing in a path, to rearrange it in order to make the path more concise [37]. Thus, the robot will not return to the same nodes more than once along this path. In addition, *insertion* is developed to make invalid path qualified by inserting nodes between unconnected nodes. Various customized operators have enlarged the field of application of GA-based method vastly.

In this study, for the multiple goals visiting task, we proposed a novel tailored genetic algorithm to find an optimal path to visit as more goals as possible. The remainder of this paper is organized as follows: in section 2, we will state the problem including the model of work environment, the multiple goals visiting task and properties of a path. In section 3, tailored genetic algorithm is described in detail. Then, simulations and analysis of results are conducted in section 4. Finally, conclusion and future work are addressed.

2. Problem Formation

In this section, we further discuss the problem including the model of work environment and the task of multiple goals visiting. At last we introduce the properties of a path that are very important for the proposed genetic algorithm.

2.1. Work environment

We use a graph-based topological map to describe the work environment [1], which is illustrated in Figure 1.

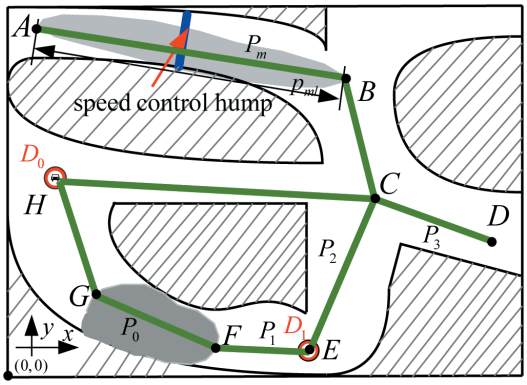

Fig. 1. Work environment.

Let V be the environmental space that includes three parts, namely, path segments, nodes connecting path segments, and obstacle regions, which are described as follows.

P: The set of path segments. In Figure 1, $P_m(e.g.,m=0,1)$ represents a path segment and for each one, four attributes are considered, i.e., path length p_{m_l} surface roughness p_{m_r}, road grade p_{m_g} and speed-control hump p_{m_h}.* Especially, the segment with shadow implies that it is a rough segment.

N: The set of nodes connecting path segments. For example, A to H are nodes connecting two or more path segments respectively. Particularly, There are some charging dock stations placed on some nodes (e.g., charging dock stations D_0 and D_1 are placed at nodes H and E respectively). In a concrete mission, if any one node is designated to visit, it is called a goal, and the position where the robot starts from is called the start point. Since each segment has two nodes, a segment can be also represented by nodes. For example, in Figure 1, segment P_0 can be also represented as P_{GF} if in one path the robot moves from node G to F, or P_{FG} from F to G.

O: The set of obstacle regions. In Figure 1, the regions with gray oblique lines represent obstacle areas. Thus we can describe the environment as $V=\{P,O,N\}$.

In previous study [1,39], we have elaborated the cost of a path segment. The cost that the robot will pay for passing each segment includes two parts: energy consumption c_e and the influence of vibration on robot body c_b. and the calculation of c_e and c_b is also deduced.Further we clarified how to computing the idle time T_{IDLE} based on c_e and c_b (refer to Ref. [1] and [39] for more details).

2.2. Optimal path planning for multiple goals visiting Task

Normally when performing regular inspecting task, the robot moves in accordance with predefined route in the environment. Occasionally, the robot may be asked to go to multiple goals to execute particular missions. To make it easy to analyze and understand, in subsections 2.2, 2.3 and section 3, we will use D_i (e.g., i=0,1) to represent charging dock and G_j (e.g., j=0,1) the goal. For example, in Figure 2, when the robot is at point S, it is commanded to visit G_1, G_2 and G_3. The robot can select the path coloured in blue to visit all the goals. Thus, the sequence of goals visited is $G_1 \rightarrow G_2 \rightarrow G_3$, for which we use $\Gamma_1 = \{S,A,G_1,G_2,D,G_3\}$ to describe the path. However, the robot may choose visiting G_3 before G_2, then the sequence becomes $G_1 \rightarrow G_3 \rightarrow G_2$ and consequently we obtain another feasible path $\Gamma_2 = \{S,G_1,G_3,G_2\}$ that is colored in red. The optimal path planning for multiple goals visiting task is to find the optimal one among all the accessible paths.

*p_{m_l}, p_{m_r}, p_{m_g} and p_{m_h} refer to p_{ml}, p_{mr}, p_{mg} and p_{mh} respectively in [1].

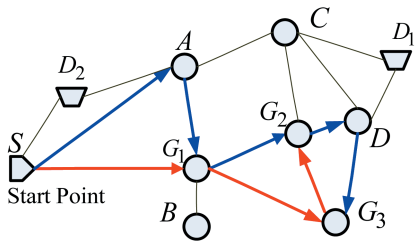

Fig. 2. Multiple goals visiting task.

The ideal situation is that the robot can visit all the goals by using the remaining energy. But in real cases, it is possible that the remaining energy is not sufficient for the robot to visit all the goals. Therefore, a compromising solution is to cut down a goal or more, and then try to find an optimal path for the visiting task. The expecting result is to make the robot visit as more goals as possible under the precondition that after arriving at the last goal the robot still has enough energy to reach one charging station. For example, in Figure 2, if the robot cannot get to G_3 when following the most energy-saving path Γ_1, it can decide not visiting goal G_3, then the optimal path is $\Gamma_1 = \{S, A, G_1, G_2\}$ and G_2 becomes the last goal to be visited. Certainly the worst situation is that the remaining energy can not support for visiting even one goal. Consequently, four cases can be obtained that are shown in Figure 3.

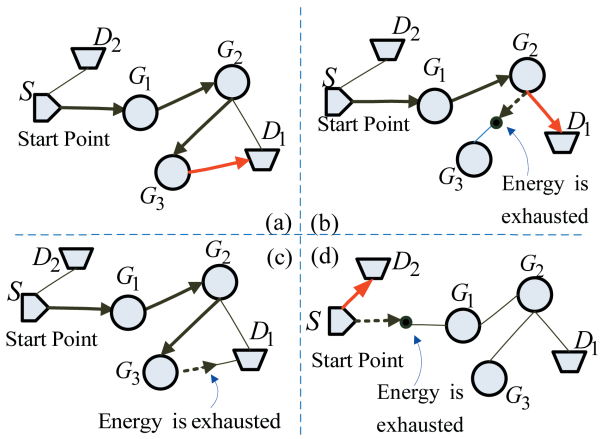

Fig. 3. Four cases of task execution.

The four situations shown in Figure 3 are described as follows.

(i) The robot can visit all the goals and the remaining energy is enough for reaching a charging station, which is shown as situation (a) in Figure 3;

(ii) The robot can just visit a portion of the goals and has to go back to one charging station. For example, in situation (b), the robot can not get to G_3 after visiting G_2 which means the robot can just visit two goals, i.e., G_1 and G_2.

(iii) The robot has sufficient energy to visit all the goals with the energy-saving path, but it cannot get to the nearest charging station from the last goal. Therefore, the robot should give up the last goal. For instance, in situation (c), the robot has to give up the last goal G_3.

(iv) The robot can not visit even one goal and has to go back to recharge. In situation (d), the robot even has not enough remaining energy to visit the first goal G_1. It should go back to charging station to recharge right now. Thus, No path is feasible for completing this task.

2.3. *Properties of a path*

We use the combination of nodes to represent a path. In this research, three basic properties of a path are obtained:

(i) A path is constituted of parts of the nodes. For example, the path colored in blue in Figure 2 can be described as $\Gamma_1 = \{S, A, G_1, G_2, D, G_3\}$. This path is constituted by nodes S, A, G_1, G_2, D and G_3 in which G_1, G_2 and G_3 are the goals assigned.

(ii) There is no priority or constraint for the sequence of goals to be visited. For example, in Figure 2, both paths $\Gamma_1 = \{S, A, G_1, G_2, D, G_3\}$ and $\Gamma_2 = \{S, G_1, G_3, G_2\}$ are valid for visiting goals G_1, G_2 and G_3.

(iii) It is permissible for a node appearing in the sequence more than once. For instance, in Figure 4, one available path is $\Gamma = \{S, G_1, S, G_2, G_4, G_2, G_3\}$, where S and G_2 appear twice. The purpose of the first arrival at one goal is to "visit" the goal in order to perform task, and that of the other times are for going to other goals.

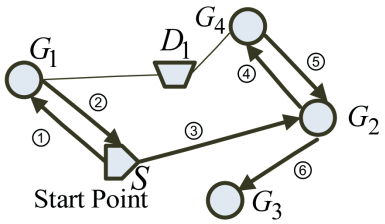

Fig. 4. A special situation.

3. Proposed Genetic Algorithm for Path Planning

In this section, we will first introduce basic genetic operators which play important role in basic GA. Then the proposed tailored genetic algorithm is introduced to solve the problem described above.

3.1. Basis of genetic operators

As an advanced stochastic search technique similar to natural evolution based on the principle of "survival of the fittest" [11], traditionally, genetic algorithm involves three basic genetic operators to simulate the adaptive process of natural systems: *Selection*, *Crossover* and *Mutation* [9].

Selection is an operator to select the survival in a set of present candidate individuals (usually being called *population*) according to the fitness value computed by the fitness function. The selected individual(s) will be kept surviving in the next generation.

Crossover is an operator adopted to reform the survival candidates. Usually, it is performed by exchanging parts of strings by use of old strings and then new strings are generated. This process derives from the natural system, in which a set of creatures creates a new set of the next generation by swapping among the creatures. Often the parts are crossed in couples of candidates selected randomly. When using this operator, one should determine a crossing rate to decide how often the selected individuals will carry out this operation.

Mutation, which means to replace one random gene of the chromosome by an arbitrary different gene, is another way to increase the diversity of population. Compared to *Crossover*, mutation rate is much smaller. Another important function of this operator is to avoid trapping in the local minima in the search space.

3.2. Tailored genetic algorithm

Based on traditional genetic algorithm, modifications are made to fit our problem. We use the combination of nodes to represent the chromosome. The fitness functions include two parts which are used to calculate energy consumption and idle time respectively. Except the basic three operators, i.e., *Selection*, *Crossover* and *Mutation*, we create three new operators: *Repair*, *Cut* and *Deletion*.

3.2.1. Chromosome

The proposed genetic algorithm uses the combination of nodes for path representation. An example of path encoding is shown in Figure 5, which is $S-A-G_1-G_2-D-G_3$. In this chromosome, S is the start point, G_1, G_2, and G_3 are three goals.

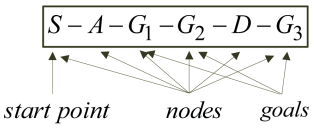

Fig. 5. An example of chromosome.

Two different chromosomes may have different length. For example, the length of the chromosome shown in Figure 5 is 6, in which 3 goals are involved. While the length of the chromosome in Figure 2, $S-G_1-G_3-G_2$, is 4, and the same 3 goals are included.

3.2.2. Path evaluation

Chromosomes are selected for reproduction through genetic operators based on the fitness function, so it is important to establish a set of criteria to evaluate the quality of a path. For each individual, we adopt three terms Ω, F_e and $F_{T_{IDLE}}$ to describe it. Ω is the number of goals involved in this chromosome. Thus, $\Omega(\Gamma)$ represents the number of goals in path Γ. F_e is the energy that the robot will spend on moving along this path, and $F_{T_{IDLE}}$ indicates the idle time induced by this path. The total energy consumption of a path

is the sum of that of each path segment, so

$$F_e = \sum_{i=1}^{h} c_e(P_i) \ . \tag{1}$$

where h is the number of segments, $c_e(P_i)$ is the energy consumption of the ith segment.

Similarly, $F_{T_{IDLE}}$ can be calculated as

$$F_{T_{IDLE}} = \sum_{i=1}^{h} T_{IDLE}(P_i) \ . \tag{2}$$

where h is the number of segments and $T_{IDLE}(P_i)$ is the idle time of the ith segment. $c_e(P_i)$ and $T_{IDLE}(P_i)$ are introduced in Section 2.1. For example, for the individual $S-A-G_1-\ G_2-D-G_3$ shown in Figure 5, we have

$$\begin{aligned} F_e = c_e(P_{SA}) + c_e(P_{AG_1}) + c_e(P_{G_1G_2}) + \\ c_e(P_{G_2D}) + c_e(P_{DG_3}) \ . \end{aligned} \tag{3}$$

$$\begin{aligned} F_{T_{IDLE}} = T_{IDLE}(P_{SA}) + T_{IDLE}(P_{AG_1}) + \\ T_{IDLE}(P_{G_1G_2}) + T_{IDLE}(P_{G_2D}) + T_{IDLE}(P_{DG_3}) \ . \end{aligned} \tag{4}$$

Define E_{rest} as the available energy the robot can utilize to perform the task, then we get

$$E_{rest} = E_{cur} - E_{low} \ . \tag{5}$$

where E_{cur} is the remaining energy the robot has when it is at the start point, and E_{low} is the threshold value of low energy.

We define e_{task} as the extra energy consumption that the robot spends on executing task at each goal, and assume that it is a fixed value, namely, $e_{task} = \tau$. For a path Γ, the total extra energy E_{task} spending on performing task is

$$E_{task} = \Omega(\Gamma) * e_{task} = \Omega(\Gamma) * \tau \ . \tag{6}$$

Thus, we get another condition that makes one path feasible, namely,

$$F_e(\Gamma) + E_{task} \leqslant E_{rest} \ . \tag{7}$$

which means that the sum of the energy spent on walking along the path and executing tasks should be less than the remaining energy.

3.2.3. Genetic operators

In proposed genetic algorithm, except the three basic operators, i.e., *Selection*, *Crossover* and *Mutation*, we create three new operators, i.e., *Repair*, *Cut* and *Deletion*.

Selection The selection process includes two steps. First, using the strategy "elitism", it will find out the best chromosome and keep it in the population in the next generation. This operation will be helpful for finding the global optimal solution. The selection is based on the fitness value. At the beginning of the algorithm, candidate solutions are generated randomly, which constitute the initial population. We assume that there exists at least one feasible path Γ that involves all the goals and its energy consumption $F_e(\Gamma)$ meets Equation 7, then in the initial population each candidate solution will include all the goals. When executing selection operation, we first check if any individual meets Equation 7. If at least one solution conforming to this condition is found, we will use $F_{T_{IDLE}}$ as the criterion to select the best one from individuals that meet this condition. Otherwise, if none is found, the selection process will utilize F_e to evaluate a path since the one having less energy consumption is more likely to be evolved to a feasible one. Subsequently, the best one that has the minimal F_e or $F_{T_{IDLE}}$ will be selected to remain in the next generation. This strategy can guarantee that the best one up to now will not be destroyed by other genetic operations and can accelerate the convergence of the algorithm.

Secondly, to keep the number of individuals in next generation unchanged, it will select out the rest stochastically. This strategy can prevent the algorithm converging to the local optimal.

Crossover *Crossover* is an efficient way to add diversity to the population. First, a crossover probability (noted as P_c) is predefined. In this operation, two parents are selected randomly and a position is selected randomly too. Then, a random probability is generated. If the probability value is less than the predefined value, the operation will go on. Otherwise, the two parents are passed to the next generation directly. The operation will end until certain

times of crossing operations are carried out.

When executing *Crossover* operation, a crossover point will be generated. Since the length of two parents may be different, the sequence number of the point will not be bigger than the length of the shorter one. Then, in the other parent, we find the corresponding node and its sequence number of the first appearance. If the other one has the same node, exchange the latter parts of the two parents. If not, quit and restart from choosing parents.

The following is an example of *Crossover* operation. First, two parents are selected:
Parent 1: $S-G_1-G_3-G_2$
Parent 2: $S-A-G_1-G_2-D-G_3$
If node G_1 is selected as the position for exchanging, then we get the offspring after crossing:
Child 1: $S-G_1-G_2-D-G_3$
Child 2: $S-A-G_1-G_3-G_2$
After crossing, the two children are put into the population of next generation.

Mutation In *Mutation* operation, a position is randomly chosen and the node at this position is replaced with a different node. Like "*Crossover*", a mutation rate P_m is also predefined. *Mutation* is served as a key role to diversify the solution population. Therefore, it is not necessary that a solution is better after mutating. After mutating, this node may not be connected directly with the two nodes before and after. For example, if node A in path $S-A-G_1-G_2-D-G_3$ shown in Figure 2 is chosen to mutate, and changes to C, then, this individual becomes $S-C-G_1-G_2-D-G_3$. However, as seen in Figure 2, nodes S and C, and C and G_1 are not connected directly, which means that the individual after mutation is not a feasible solution. Even so, it has made the population diversified, and the following operator *Repair* can make it feasible.

Repair When executing genetic operators, some infeasible paths may be generated. For instance, after mutation, individual $S-A-G_1-G_2-D-G_3$ becomes $S-C-G_1-G_2-D-G_3$, while nodes S and C and C and G_1 are not connected directly. When this happens, we will use *Repair* operator to solve this problem. The practical way is inserting some suitable nodes between the two nodes.

Take $S-C-G_1-G_2-D-G_3$ as an example. When executing repair operation, we first check if this individual is feasible by examine every two adjacent nodes. If at a position, the node and the next node are not connected directly, then, this operator will try to add some nodes between them in order to make the two nodes connected reasonably. In the above example, the nodes S and C may be inserted by node A. and then C and G_1 may be inserted by nodes G_2 or A, which is decided randomly. If A s selected, then the individual is repaired to be $S-A-C-A-G_1-G_2-D-G_3$, and if G_2 is selected, it will becomes $S-A-C-G_2-G_1-G_2-D-G_3$. No matter whichever is chosen, the result is that the path becomes feasible at last.

It is notable that after mutation or crossover, the number of goals may change. If an individual should have k goals but just involve $k'(k'<k)$ goals now, the repair operator will try to repair this string by adding nodes after the last node until the other $k-k'$ goals appear at least one time.

Cut In a chromosome, it is allowed that any node appears more than one time. But the unnecessary reduplication must be avoided. For example, in string $S-A-C-A-G_1-G_2-D-G_3$ obtained after repairing, node A appears twice and between them there is no goal. It can be regarded as that between the two times arriving at A, the intention is not for going to any goal. So, the sequence $C-A$ is meaningless and it needs to be cut. Finally, this string becomes $S-A-G_1-G_2-D-G_3$. So, the *Cut* operator is to do such things that cutting the unmeaning sequences existing in each individual.

However, the reduplication does not include the situation that a goal exists between the same two nodes. For instance, in chromosome $S-A-C-G_2-G_1-G_2-D-G_3$, G_2appears twice. But between them there is another goal G_1 which indicates that the purpose of arriving at G_2 for the second time is for visiting another goal. Thus, the second time passing G_2 is meaningful.

From the above analysis, two criteria can be used as judging whether a chromosome needs *Cut* operation:

(i) If a node that is not a goal appears twice, and no goal is between them, then this sequence after the node can be cut. For example, the sequence $A-C-A$ appearing in path $S-A-C-A-G_1-G_2-D-G_3$ can be cut to A.

(ii) In an individual, for each goal, only the first appearance is regarded as a goal. The other times of appearance will be regard as a common path node, which implies that the purpose of passing the goal for the second or third time is for reaching another goal. Take $S-A-G_1-A-\cdots-B-G_1-B-G_3$ as an example, in which the robot reaches G_1 twice. Only the first time is for visiting this goal. This is why we cannot execute *cut* operation on the part $A-G_1-A$ On the contrary, the second time when it appears, it will be deemed as a common node. Therefore, $B-G_1-B$ should be cut to be B according to rule i.

Deletion In a chromosome, when all the goals have appeared once, the remainder of the string is not necessary and should be deleted. Assume that the string is admitted to include k ($0<k\leqslant N$, where N is the number of goals assigned at the beginning) goals, then if $k'(k'>k)$ different goals appear in the string, we should delete the string after the kth goal.

When it is confirmed that the robot has no sufficient energy to visit all the goals, it will attempt to reduce one goal and go on searching the optimal path. For example, in the task of visiting goals G_1, G_2 and G_3, if it is found that the robot has no sufficient energy to visit all the three goals, then the robot will try visiting two goals. Thus, the path $S-A-C-G_2-G_1-G_2-D-G_3$ will become $S-A-C-G_2-G_1$ after performing *Deletion* operation.

3.2.4. *Execution of proposed tailored genetic algorithm*

Compared to traditional genetic algorithm, variations are made in the execution of proposed algorithm. It is cyclically executed, where the criterion for deciding the circulation is to check if any path satisfying the energy constraint. In each cycle, the evolution process is similar to the basic GA. In Figure 6, the execution of proposed GA is illustrated.

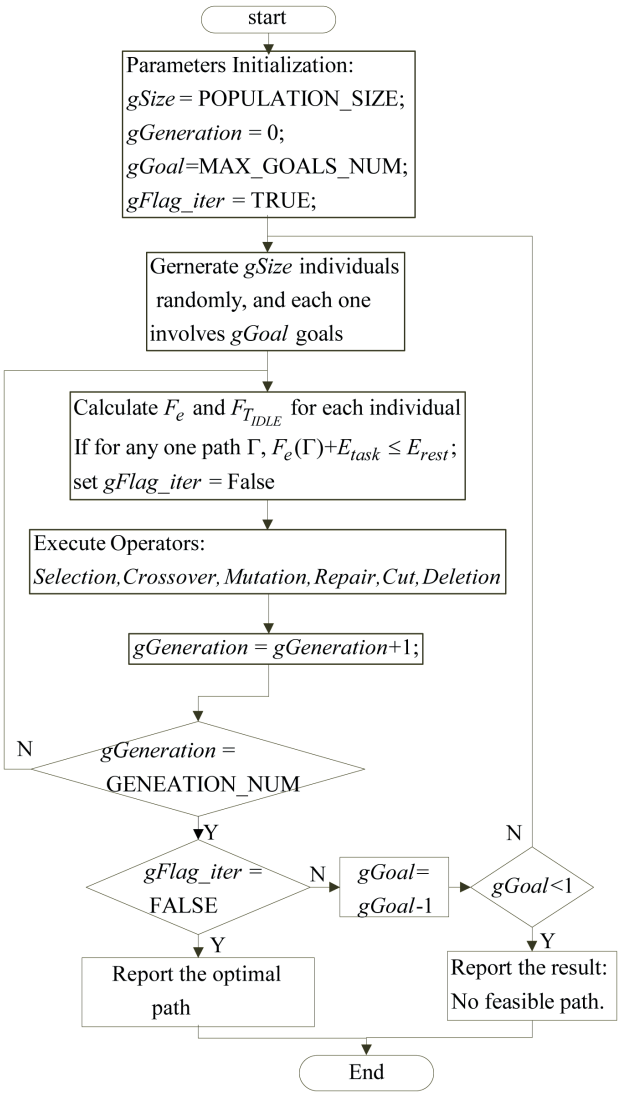

Fig. 6. Execution of the proposed genetic algorithm.

In Figure 6, *gSize* is the population size, and *gGeneration* is the number of generations. *gGoal* indicates how many goals are involved in each individual, and $gFlag\_iter$ is used for determining if another circulation of performing the basic genetic algorithm is needed. In the entire procedure, *gSize* keeps being $POPULATION\_SIZE$ that is constant. On the contrary, *gGoal* may reduce

if $gFlag\_iter$ remains $TRUE$ when beginning another cycle. At the beginning, the value of $gGoal$ is MAX_GOALS_NUM which is the total number of goals to be visited. If $gFlag\_iter$ remains $TRUE$ when one cycle is over, $gGoal$ may reduce by one until it decreases to zero, which means trying to find an optimal path with one less goal. If in one cycle, $gFlag\_iter$ turns into $FALSE$, the whole procedure will finish when $gGeneration$ increases to $GENERATION\_NUM$ which is a predefined constant.

In each cycle, the basic tailored genetic algorithm is conducted to search the optimal path of corresponding number of goals. In the end, if $gGoal$ is not smaller than one, then one optimal path is obtained, otherwise, it will report that no feasible path is available. This mechanism will guarantee that the algorithm obtains an optimal path involving as more goals as possible if at least one feasible path exists.

4. Simulation Studies

In this section, simulations are implemented to examine our proposed tailored genetic algorithm.

4.1. Simulations and results

We use the topological map shown in Figure 7 in simulations, which is built in previous work (see [1]). There are 23 path segments and 17 nodes segments in the environment. In addition, the attributes of each segment are also listed in [1]. In simulations, parameters in the proposed genetic algorithm are set as follows: $POPULATION\_SIZE$= 30, and $GENERATION\_NUM$= 100. Crossover rate P_c = 0.9 and P_m = 0.001. To simplify the computation, we assume $e_{task} = \tau = 0$, then $E_{task} = 0$. Therefore, Eq. (7) becomes $F_e(\Gamma) \leqslant E_{rest}$. Furthermore, we assume that E_{rest} is a constant. In the following simulations, we set $E_{rest} = 0.17V$.

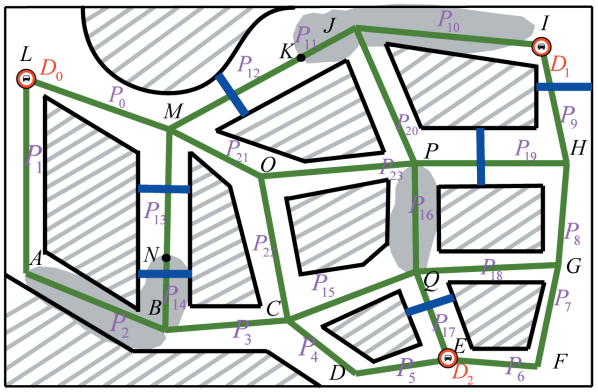

Fig. 7. Topological map of environment.

4.1.1. Simulation I

In this test, node A is set as the start point, and the goals are C, H and M. The energy consumption and idle time of the best individual in each generation are shown in Figure 8. It is obtained from the result that the optimal solution comes out in the 18^{th} generation. The optimal path is $A-B-C-O-M-O-P-H$. Its energy consumption $F_e = 0.1511V$ and idle time $F_{T_{IDLE}}$ = 1022.1330s. Since $F_e < E_{rest}$ = 0.17V, this path is feasible, and all goals can be visited. Further more, the order of visiting is C, M and H. The computational time is measured to be 151ms.

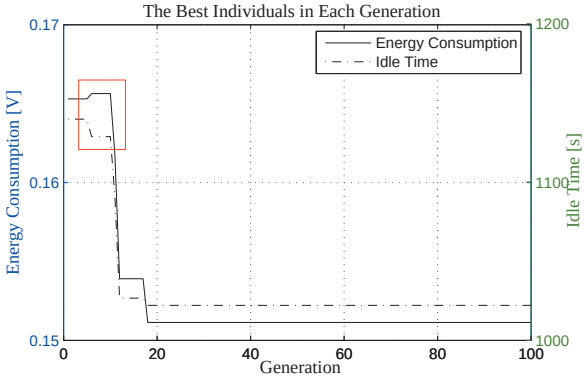

Fig. 8. Result of simulation I.

In addition, there is a notable situation shown in the red rectangle in Figure 8. For the best individ-

ual found in the 6^{th} generation, its idle time is less than that of the 5^{th} generation, while the energy consumption is more. This is because in the former 5 generations, we have found one path satisfying the condition of energy constraint, then the minimal idle time will be employed as principal for selecting the optimal path.

Table 1. Details of Best Individuals in Each Generation

Generation	Best individual	$F_e(s)$	$F_{T_{IDLE}}(s)$
1-5	$A-B-C-B-N$ $-M-O-P-H$	0.1653	1139.6626
6-10	$A-L-M-N-B$ $-C-Q-G-H$	0.1656	1128.5902
11	$A-L-M-O-C$ $-O-P-H$	0.1617	1090.5179
12-17	$A-L-M-O-C$ $-Q-G-H$	0.1540	1026.6969
18-100	$A-B-C-O-M$ $-O-P-H$	0.1511	1022.1323

For sake of further understanding, we list out the concrete data of energy consumption and idle time of the best one in each generation in Table 1, the energy consumption of the best one in the first generation is $0.1653V$ that is less than $0.17V$. So, from the second generation, the best individual is selected from the candidates that satisfying energy condition by using idle time as criterion.

4.1.2. *Simulation II*

In this simulation, we set A as the start point, and the goals are H, N, O and Q. One result is shown in Figure 9. In the first 29 generations, energy consumption keeps more than $0.17V$, thus, during this period, the criterion for selecting optimal individual in each generation is energy consumption. In the 30^{th} generation, we get the optimal path that is $A-B-N-M-O-C-Q-G-H$. Its energy consumption is $0.1646\,V$ and idle time is $1123.5112\,s$. The result shows that the robot also has sufficient energy visiting all the goals, and the order is N, O, Q and H. The computational time is $157ms$.

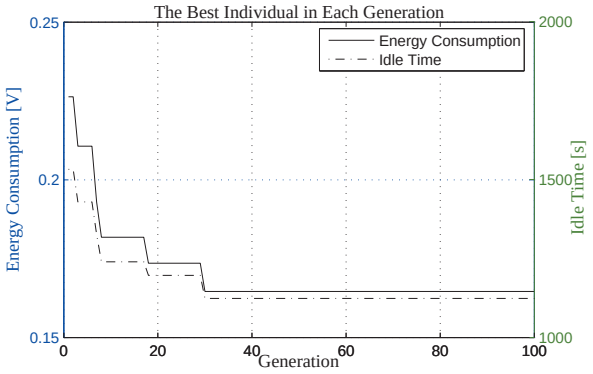

Fig. 9. Result of simulation II.

4.1.3. *Simulation III*

In this simulation, A is the start point, and goals are E, H, N and O. We show the result in Figure 10.

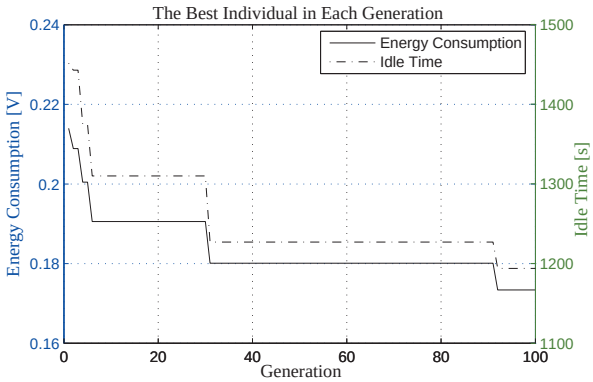

Fig. 10. Result of simulation III.

From Figure 10, we get the optimal solution in the 92^{nd} generation in the first cycle. The energy consumption F_e is $0.1743V$ and idle time is $1193.8413s$. Because F_e exceeds the remaining available energy $0.17V$, this path is not feasible and we will continue searching another optimal path with one goal less. With this strategy, a new optimal path that both satisfies requirement of energy consumption and has the minimum idle time is generated: $A-B-N-M-O-P-H$. Its energy consumption is $0.1274V$ and idle time is $886.9106s$. The energy consumption is less than $0.17V$, however, it just allow the robot to visit three goals which are N,

O and H. The total computational time is measured to be 282ms.

4.2. Analysis of simulation results

In the three simulations above, we implement our proposed tailored genetic algorithm to find the optimal path for multi-goal visiting task and finally optimal solutions are obtained. In the following we will discuss about the similarity and difference between each case and evaluate the proposed genetic algorithm based on simulation results.

(i) As the genetic algorithm itself is a kind of stochastic, evolutionary search method, the optimal solution obtained at the end may not be the global optimal one truly.

(ii) In the three cases, the speed of converging to the optimal solution is different. For example, the optimal one appears in the 18^{th} generation in simulation I, while it is obtained in the 30^{th} generation in simulation II. Moreover, in simulation III, in the first cycle, the optimal one is found in the 92^{nd} generation, and is proven to be not feasible.

(iii) The computational time in case I and II are 151ms and 157ms respectively, and only one cycle of evolution is executed. In simulation III, two cycles are performed and 282ms is needed. It shows great efficiency in computation. However, in the simple environment there are just 17 nodes. To verify the timeliness and efficiency, a more complicated environment needs to be established and more simulations are required.

(iv) Generally, when using GA method, the stop condition can be either that the best solution keeps unvaried for certain number of generations, or that the current maximum generation is exceeded [40]. In proposed genetic algorithm, the latter is adopted. However, in reality, both can not ensure the final solution is truly the optimal one, and therefore it is uncertain that which one is better absolutely. As an example, in simulation III, the solution generated firstly in the 30^{th} remains the best one in the following 62 generations. If we use the former stop criterion, and set the maximum generation to be 50 or 60, this solution will be regarded as the final optimal path. However, it is soon replaced by a better solution. Furthermore, if we set the maximum generation is 90, we also can not get the better solution that comes out soon.

(v) In all the simulations, parameter E_{rest} is considered as a constant for simplifying the problem. Nevertheless, for different paths, the last goal reached may not be the same, this value will be different. Actually, this value should be the minimum energy the robot needs to move to one charging station from the last goal. Hence, in practice, it is wiser to use different values of E_{rest} rather than the same fixed value.

5. Conclusion

We have proposed a novel tailored genetic algorithm to solve the problem of optimal path planning for multiple goals visiting task. Aiming at the particularity of the problem, special form of chromosome is used to represent the path and customized genetic operators are development. The effectiveness of the method is verified by simulations. Furthermore, through analysis of simulation results, evaluation on our proposed method is addressed, which is useful for wider implementation in various circumstances. Future work will be carried out to perfect this method by considering parameters more realistically, and to execute more simulations with more complicated environment. At last, we will implement the proposed algorithm on real systems.

Acknowledgements

This research was sponsored by the Key Project of Science and Technology Committee of Chongqing (CSTC, 2009AB2139).

Reference

1. F. Liu, S. Liang, and X. D. Xian,"Determination of an optimal return-path on road attributes for mobile robot recharging," *Intl. J. of Advanced Robotic Systems*, **8(5)**, 83–92 (2011).

2. O. Madsen, C. B. Sϕrensen, R. Larsen, L. Overgaard, and N. J. Jacobsen,"A system for complex robotic welding," *Industrial Robot: An Intl. J.*, **29(2)**, 127–131 (2011).
3. R. Thrapp, C. Westbrook, and D. Subramanian,"A system for complex robotic welding," *Proc. of IEEE Intl. Conf. on Robotics and Automation*, 2065–2071 (2001).
4. J. J. Rainer, R. Galan, B. M. Al-Hadithi, and A. Jimenez,"Apg: An intelligent automatic generator of presentations for tour-guide robots," *Intl. J. of Computational Intelligence Systems*, **4(4)**, 446–455 (2011).
5. P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *Intl. J. of Physical Sciences*, **7(9)**, 1314–1320 (2012).
6. N. Sariff and N. Buniyamin, "An overview of autonomous mobile robot path planning algorithms," *in 4th Student Conf. on Research and Development*, 183–188 (2006).
7. M. Yuan, S. A. Wang, C. Wu, and N. Chen, "A novel immune network strategy for robot path planning in complicated environments," *J. of Intelligent & Robotic Systems*, **60(1)**, 111-131 (2010).
8. M. McNaughton and C. Urmson, "Fahr: Focused a* heuristic recomputation," *Intl. Conf. on Intelligent Robots and Systems*, 4893–4898 (2009).
9. T. Shibata and T. Fukuda, "Intelligent motion planning by genetic algorithm with fuzzy critic," *Intl. Symposium on Intelligent Control*, 5–10 (1993).
10. O.Wongwirat and A. Anuntachai, "Searching energy-efficient route for mobile robot with ant algorithm," *The 11th Intl. Conf. on Control, Automation and Systems*, 1071–1075 (2011).
11. S. X. Yang and Y. Hu, "A knowledge based GA for path planning of multiple mobile robots in dynamic environments," *IEEE Intl. Conf. Robotics and Automation*, 71–76 (2007).
12. T. Zheng and P.Wang, "Priority based dynamic multiple robot path planning," *The 2nd Intl. Conf. on Autonomous Robots and Agents*, 373–378 (2004).
13. L. E. Parker, "Path planning and motion coordination in multiple mobile robot teams," *Encyclopedia of Complexity and System Science*, 1–24 (2009).
14. T. Hellstrom and O. Ringdahl, "Real-time path planning using a simulator-in-the-loop," *Intl. J. of Vehicle Autonomous Systems*, **7(1/2)**, 56–72 (2009).
15. R. Iraji, "Robot path planning using wavefront approach with wall-following," *THE 2nd IEEE Intl. Conf. on Computer Science and Information Technology*, 417–420 (2009).
16. H. F. Wang and Y. Z. Yang, "Time-optimal trajectories for a car-like robot," *Automatica*, **34(4)**, 445–452 (2008).
17. Z. Sun and J. H. Reif, "On finding energy-minimizing paths on terrains," *IEEE Trans. on Robotics*, **21(1)**, 102–114 (2005).
18. S. Liu and D. Sun, "Optimal motion planning of a mobile robot with minimum energy consumption," *Intl. Conf. on Advanced Intelligent Mechatronics (AIM2011)*, 43–48 (2011).
19. Y. Mei, Y. hsiang Lu, Y. C. Hu, and C. S. G. Lee, "Energy-efficient motion planning for mobile robots," *Intl. Conf. on Robotics and Automation*, 4344–4349 (2004).
20. E. Lobaton, J. Zhang, S. Patil, and R. Alterovitz, "Planning curvature-constrained paths to multiple goals using circle sampling," *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 1463–1469 (2011).
21. F. Liu, S. Liang, X. D. Xian, and H. B. Bi, "Optimal path planning for mobile robot in consideration of road attributes," *ICIC Express Letters*, **6(1)**, 281–287 (2012).
22. N. Sedaghat, "Mobile robot path planning by new structured multi-objective genetic algorithm," *The 3rd Intl. Conf. of Soft Computing and Pattern Recognition (SoCPaR)*, 79–83 (2011).
23. F. Castillo, L. Trujillo, and P. Melin, "Multiple objective genetic algorithms for path-planning optimization," *Soft Computing*, **11**, 269–279 (2007).
24. G. Capi, "Multiobjective evolution of neural controllers and task complexity," *IEEE Trans. on Robotics*, **23(6)**, 1225–1234 (2007).
25. E. Masehian and D. Sedighizadeh, "A multi-objective pso-based algorithm for robot path planning," *IEEE Intl. Conf. on Industrial Technology(ICIT)*, 465–470 (2010).
26. R. T. Marler and J. S. Arora, "Survey of multiobjective optimizationmethods for engineering," *Multidisc Optim*, **26**, 369–395 (2004).
27. G. Avigad and K. Deb, "The sequential optimization constraint multi-objective problem and its applications for robust planning of robot paths," *IEEE Congress on Evolutionary Computation*, 2101–2108 (2007).
28. O. M. Garcia MAP, O. Castillo, R. Sep$'$*u*lveda, and P.Melin, "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation," *Applied Soft Computing*, **9(3)**, 1102–1110 (2009).
29. F. Ahmed and K. Deb, "Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms," *Soft Computing*, 1–17 (2012).
30. A. Akbarimajd and A. Hassanzadeh, "Autonomously implemented versatile path planning for mobile robots based on cellular automata and ant colony," *Intl. J. of Computational Intelligence Systems*, **5(1)**, 39–52 (2012).
31. Holland, "Adaptation in Natural and Artificial Systems," *Ann Arbor: University of Michigan Press* (1975).

32. A. Konak, D. W. Coit, and A. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability Engineering*, **91(9)**, 992–1007 (2006).
33. I. AL-Taharwa, A. Sheta, and M. Al-Weshah, "A mobile robot path planning using genetic algorithm in static environment," *J. of Computer Science*, **4(4)**, 341–344 (2008).
34. G. Nagib and W. Gharieb, "Path planning for a mobile robot using genetic algorithms," *Proc. of the Intl. Conf. on Electrical, Electronic and Computer Engineering (ICEEC'04)*, 185–189 (2004).
35. J. Tu and S. X. Yangt, "Genetic algorithm based path planning for a mobile robot," *Intl. Conf. on Robotics and Automation*, 1221–1226 (2003).
36. M. Mansouri, M. A. Shoorehdeli, and M. Teshnehlab, "Integer ga for mobile robot path planning with using another ga as repairing function," *Proc. of the IEEE Intl. Conf. on Automation and Logistics*, 135–140 (2008).
37. Z. Yao and L. Ma, "A static environment-based path planning method by using genetic algorithm," *Intl. Conf. on Computing, Control and Industrial Engineering*, 405–407 (2010).
38. F. Ahmed and K. Deb, "Multi-objective path planning using spline representation," *IEEE Intl Conf. on Robotics and Biomimetics (RO- BIO)*, 1047–1052 (2011).
39. F. Liu, S. Liang, and X. D. Xian, "Corrigendum to determination of an optimal return-path on road attributes for mobile robot recharging," *Intl. J. of Advanced Robotic Systems*, **9(5)**, 1–3 (2012).
40. E. Zitzler, M. Laumanns, and S. Bleuler, "A tutorial on evolutionary multiobjective optimization," *Evolutionary Computation*, **535(5)**, 3–37 (2004).