applied sciences

MDPI

Article

A Re-Entry Path Planning Method for Service Robots Based on Dynamic Inver-Over Evolutionary Algorithm

Yong Tao [1,2,*], Chaoyong Chen [3], Tianmiao Wang [1,2], Youdong Chen [1], Hegen Xiong [3], Fan Ren [1] and Yu Zou [3]

1 School of Mechanical Engineering and Automation, Beihang University, Beijing 100191, China; itm@buaa.edu.cn (T.W.); chenyd@buaa.edu.cn (Y.C.); 17801034518@163.com (F.R.)
2 Beijing Advanced Innovation Center for Biomedical Engineering, Beihang University, Beijing 100083, China
3 The Key Laboratory of Metallurgical Equipment and Control of Education Ministry, Wuhan University of Science and Technology, Wuhan 430081, China; 15827624228@163.com (C.C.); xionghegen@126.com (H.X.); zouyu194@163.com (Y.Z.)
* Correspondence: taoy@buaa.edu.cn

Received: 28 November 2019; Accepted: 27 December 2019; Published: 31 December 2019

Abstract: A re-entry path planning method in omitting areas for service robots is suggested based on dynamic Inver-Over evolutionary algorithms after the robot automatically avoids obstacles. The complete coverage path planning is researched for cleaning service robots. Combined with features of dynamic travelling salesmen problem (DTSP), a local operator is employed for the path planning to enhance real-time dynamic properties of the Inver-Over algorithm. The method addresses the path planning problem that a number of cells undergo dynamic changes over time under work environment of cleaning robots. With simulations and experiments performed, it is discovered that the average relative error is 2.2% between the re-entry path planning and the best path, which validates the effectiveness and feasibility of the method.

Keywords: cleaning service robot; re-entry path planning; DTSP; dynamic Inver-Over evolutionary algorithm

1. Introduction

As technologies advance and society develops, cleaning service robots have gradually become an integral part of people's everyday life [1]. Capable of moving and avoiding obstacles automatically, these robots independently perform cleaning tasks without the need for human supervision. Complete coverage path planning is significant to cleaning service robots. In addition, to facilitate such planning, the robots are required to travel to all accessible points within an enclosed areas under certain motion control strategies [2].

Over the past two decades, complete coverage path planning algorithms have attracted an increasing attention from domestic and foreign scholars. Choset [3] proposed a classification of these algorithms into "online" and "offline" algorithms. In [4–11], some feasible methods are presented, such as cell decomposition, grid decomposition, spanning tree covering and random collision methods, whereas these methods are only suited to uninhabited areas, as they give no consideration to dynamic obstacles of environment. Under uncertain dynamic environments, there are a variety of intelligent methods such as neural network method [12,13] and genetic algorithm [14,15]. Pham and Lam [16], who integrated knowledge reasoning with breadth-first search, put forward a novel method to find the optimal path for a cleaning robot under dynamic environment. Zhou et al. [17] came up with a complete coverage path planning algorithm premised on rolling window and distance transform. The cleaning robot detected the local dynamic environment and updated local maps with the assistance

of a sensor. Subsequently, the coverage path was determined inside the rolling window by distance transform. After the robot traversed workspace, some cells are omitted for avoidance of dynamic obstacles. It is a necessity to traverse these cells over again. Qiu et al. [18] proposed a method that works based on bio-inspired neural network and heuristic search. The robot is capable to traverse the entire path to independently avoid obstacles. After the robot reached the terminal point, the area where obstacles were encountered during motion was identified by the planning algorithm for a minimum tangential path, so as to make a new path plan.

The cells omitted for autonomous obstacle avoidance under dynamic environment, and the related solutions were not referred to in these papers [12–16], which couldn't be overlooked for commercial cleaning service robots. In [17], a robot performed global search and traversed within certain area. In [18], a robot traversed in inverted order based on the information on omitting cells. In all these studies, optimal traversal sequence of omitting cells failed to be taken into consideration for path planning. In case that a cleaning robot operates under a complicated dynamic environment such as shopping malls and office buildings, it is inevitable for many cells to be omitted. Under this circumstance, unnecessary energy consumption will occur if simple traversal methods are employed. It is conducive to significantly improving work efficiency for robots by planning the optimal route for cells to traverse. Therefore, a re-entry path planning method for cleaning robots is suggested to find out the optimal traversal sequence for omitting cells of cleaning robots.

DTSP means that the shortest path across all cities can be determined invariably despite constant changes in urban position and scale. As demonstrated by comparisons, re-entry path planning for robots was similar to DTSP. When a cleaning robot traverses the workspace over again, there is a possibility for the omitting cells to be occupied by dynamic obstacles, which conforms to the dynamic characteristics of DTSP. Optimization algorithms for static TSP include ant colony algorithm [19], genetic algorithm [20], neural network algorithm [21] and particle swarm algorithm [22]. Evolutionary algorithms simulating biological evolution of the nature could perform excellently in solving TSP due to their generality and scalability. As revealed by the relevant studies [23–27], the Inver-Over evolutionary algorithm is identified as the most efficient evolutionary algorithm for solving TSP. Huang et al. [28] demonstrated that evolutionary algorithm is among the best algorithms to solve DTSP through convex polygon cutting, flow shop scheduling and gem cutting. Zhou et al. [29] suggested an Inver-Over DIOEA and proposed some standards to assess performances of DTSP algorithms. Li et al. [30] raised an improved gene pool-based Inver-Over evolutionary algorithm. The gene pool stored individuals' information and the dynamic elastic operator was used to conduct local search. These two methods significantly enhanced the efficiency of DTSP algorithm. Nevertheless, the aforementioned studies were limited to DTSP resulting from changing urban positions. They failed to ascertain whether their algorithms were effective for the re-entry path planning for cleaning robots where urban positions were kept unchanged, but the quantity of cities varied.

In this paper, a local dynamic operator is presented. A re-entry path planning method based on dynamic Inver-Over evolutionary algorithm is proposed to plan paths on a real-time basis while the number of cells varies. In Section 2, the re-entry path planning is suggested to construct digital models for DTSP and kinematic models for robots. In Section 3, a dynamic Inver-Over evolutionary algorithm is put forward. In Section 4, the experimental results of re-entry path planning with dynamic Inver-Over evolutionary algorithms are indicated. In Section 5, the conclusion is drawn and future work is indicated.

2. Problem Description and Mathematical Models

After investigation into the cells that a cleaning robot missed cleaning after it independently avoided obstacles during its work, re-entry path planning is proposed in this paper. As demonstrated by comparisons, re-entry path planning is within the scope of dynamic travelling salesman problem. The digital models for DTSP are constructed. The kinematic model is performed for differential-drive cleaning robots.

2.1. Problem Description

Problems arise with robots which automatically move for cleaning floors. Under dynamic environment conditions, cleaning robots are required to execute the complete coverage path plan within the designated area while avoiding obstacles automatically. As there are some dynamic obstacles, it is inevitable for the robot to produce some uncleaned cells, as illustrated in Figure 1. The robot starts cleaning on the rectangular map. It will automatically avoid obstacles and record corresponding cells which have yet to be cleaned when it encounters pedestrians or other dynamic obstacles. At the end of each travel across the map, a total of 14 cells are left uncleaned. Then, the robot has to track back and clean the omitting cells. How to plan the shortest path so that the robot can traverse all omitting cells is significant to backtracking.

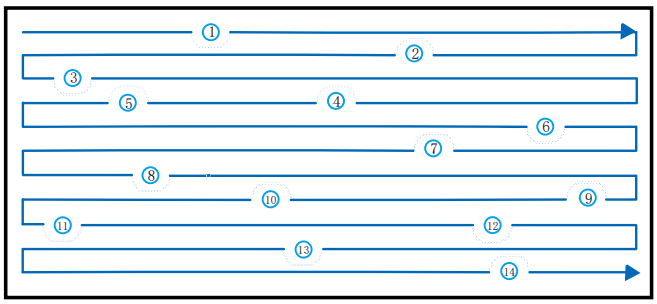

Figure 1. Schematic diagram of missing cells of cleaning robots.

The omitting cells are either occupied or free. When cells are occupied, it suggests that there are dynamic obstacles inside the cells. If they are free, the robot is allowed to clean it. Robot sensors monitor the status of omitting cells on a real-time basis, while the number of free omitting cells keeps changing with the environment. To unify the optimization of backtracking combinations for cleaning robots, in this paper, the re-entry path planning with a view to plan the shortest path across all omitting cells during real-time planning under changing environment is proposed.

Travelling salesman problem (TSP) is primarily aimed at determining the shortest path across all cities. In conventional TSP, the size of cities and distance between cities are fixed. In a variety of different practical combinations, the size of cities dynamically changes over time. Thus, DTSP is obtained. The re-entry path planning for cleaning robots is specific to DTSP, where the size of cities varies over time.

2.2. Mathematical Models for DTSP

The difference between DTSP and TSP lies in dynamic cost (distance) matrix, or distance matrix. It is expressed as follows:

$$D(t) = \left\{d_{ij}(t)\right\}_{n(t)\times n(t)} \tag{1}$$

where, the $d_{ij}(t)$ represents the distance between cities c_i and c_j, t is time. Where, the number of cities $n(t)$ and distance matrix vary over time. The DTSP aims to determine a minimum cost route across all cities at any time point t.

The model for DTSP is written as follows:

$$min\ d(T(t)) = \sum_{j=1}^{n(t)} d_{T_j,T_{j+1}}(t) \tag{2}$$

where, $T(t)$ is affiliated to the set $\{1, 2, \ldots, n(t)\}$.

In respect of the re-entry path planning method for cleaning robots, the omitting cells are urban nodes of DTSP. The distance is static between omitting cells, whereas the number of free omitting cells varies over time. Therefore, the re-entry path planning is identified as being within the scope of DTSP.

2.3. Kinematic Models for Cleaning Robots

At the bottom of cleaning robots, the wheels move with differential drive. As shown in Figure 2, there are five turning wheels on the chassis, including two middle drive wheels mounted on the same shaft and each of them can independently move forward or backward. The remained three wheels are universal wheels to offer support to vehicle bodies, with no constraint imposed. It is necessary to determine the relationships between speed of left/right wheels and travelling speed of robots. This part derives and analyses the kinematic model of robots. It's noteworthy that skid and slip phenomena in cleaning robots steering process are considered negligible in the paper.

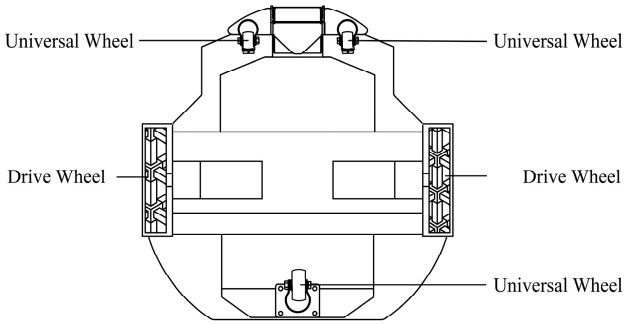

Figure 2. Chassis of cleaning robots.

The kinematic model of differential-driven wheeled robots is indicated in Figure 3.

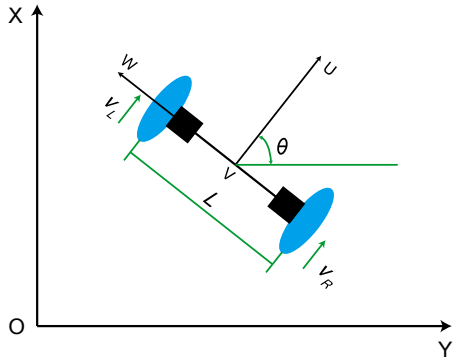

Figure 3. Kinematic model of cleaning robots.

To illustrate the position and attitude of robots under the practical work environment, the global coordinate (XOY) and local coordinate (UVW) are created, respectively. The V represents the midpoint between two drive wheels. The L indicates spacing between the two drive wheels. The linear velocity of left and right wheels denotes V_L an d V_R respectively. The θ refers to course angle of robots. It is assumed that coordinate of point V is (x, y) and the vector $[x\ y\ \theta]$ T composed of the coordinate is robot attitude. In this case, it is calculated that:

$$\begin{cases} \dot{x} = \frac{(V_L+V_R)\cos\theta}{2} \\ \dot{y} = \frac{(V_L+V_R)\sin\theta}{2} \\ \dot{\theta} = \frac{V_R-V_L}{L} \end{cases} \quad (3)$$

Equation (3) is simplified as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{\cos\theta}{2} & \frac{\cos\theta}{2} \\ \frac{\sin\theta}{2} & \frac{\sin\theta}{2} \\ -\frac{1}{L} & \frac{1}{L} \end{bmatrix} \begin{bmatrix} V_L \\ V_R \end{bmatrix} \tag{4}$$

It is observed that linear velocity and angular velocity are respectively as follows at point V, and the central position of the robot:

$$v = \sqrt{\dot{x}^2 + \dot{y}^2} = \frac{V_L + V_R}{2} \tag{5}$$

$$\omega = \dot{\theta} = \frac{V_R - V_L}{L} \tag{6}$$

Therefore, the forward speed of the robot can be obtained based on speed of left and right drive wheels, and vice versa. After kinematic models are constructed for the robot, the robot is subjected to control by sending information about speed of left and right wheels. The robot is capable of following the predetermined speed and trajectory.

3. Re-Entry Path Planning for Robots Based on Dynamic Inver-Over Evolutionary Algorithm

A local dynamic operator is designed and integrated into the static TSP algorithm. Additionally, a dynamic Inver-Over evolutionary algorithm for re-entry path planning is suggested. With favorable dynamic performances, this algorithm is conducive to tracking environmental changes in real time. As illustrated in Figure 4, the dynamic Inver-Over evolutionary algorithm follows the six steps as follows:

Step 1: Initialize parameters of the dynamic algorithm; enter position and number of omitting cells; set end time of the dynamic algorithm, population size (pop_size), inversion probability of Inver-Over operator (k) and total number of iterations for the static algorithm.

Step 2: Integrate cell position and quantity into the Inver-Over evolutionary algorithm for solutions, which are then saved.

Step 3: Decide if it has been the end time. If yes, the algorithm shall come to an end, or else jump to Step 4.

Step 4: Determine if number of cells changes under the environment. If yes, skip to Step 6, otherwise proceed to Step 5.

Step 5: Output the route for solving the static Inver-Over algorithm and skip to Step 3.

Step 6: Save changing information on cells under environment in the Darray; analyze data of changing cells with the assistance of dynamic Occupy operator and output the new route; then, skip to Step 3.

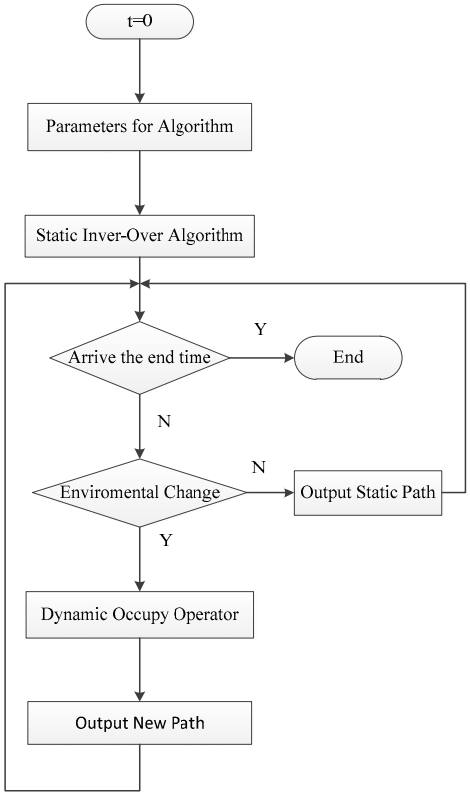

Figure 4. Flowchart of dynamic Inver-Over evolutionary algorithm.

3.1. Inver-Over Evolutionary Algorithm

In 1998, Tao proposed an evolutionary algorithm premised on the Inver-Over operator. It makes gene sequence adaptively reverse at certain probability, exhibiting the features of both mutation operators and crossover operators. The Inver-Over operators require fewer parameters than unary operators. With predominant advantages in solving mass and computing speed, they are identified as the most effective ways to solve TSP.

As indicated below, the steps in solving static TSP by Algorithm 1, especially Inver-Over operator, are described more thoroughly. A city is chosen on a random basis at a low probability (k) for inversion to increase population diversity. If $rand() > k$, a city (C) is selected at random in the male parent. Subsequently, another male parent (S) is taken and the next city of C in S' is designated to be C'. Then, C and C' in S are inverted. The original path with a new path is replaced if the new one obtained from reversion is shorter than the original one, otherwise the original path shall continue to be used. In this case, the Inver-Over operator is similar to crossover operator, which is because some cities of the second individual appear in posterity.

The Inver-Over operator can be executed in a highly efficient way, as this operator is associated with crossover and mutation. By continuously reversing gene segments, the operator is effective in improving population diversity, expanding search fields and significantly increasing the rate of convergence for algorithms.

Apparently, the properties possessed by the basic static TSP algorithm are extremely significant to the properties of DTSP algorithms. The Inver-Over evolutionary algorithm is a fairly effective static TSP algorithm. A local dynamic operator is proposed in this paper to make the Inver-Over evolutionary algorithm dynamic.

Algorithm 1 Inver-Over ()

Input: locations of cities
Output: static optimal route
1: random initialization of the population P
2: **while** not satisfied termination condition **do**
3: **for** each individual S_i in P **do**
4: $S' = S_i$
5: select (randomly) a city C from S'
6: **while** TRUE **do**
7: **if** ($rand$ () <= k)
8: select the city C' from the remaining cities in S'
9: **else**
10: select (randomly) an individual from P
11: assign to C' the 'next' city to the city C in the selected individual
12: **end if**
13: **if** the next or previous city of city C in S' is C'
14: **exit** from repeat loop
15: inverse the section from the next city of city C to city C' in S'
16: $C = C'$
17: **end while**
18: **if** $length(S')$ <= $length(S_i)$
19: $S_i = S'$
20: **end for**
21: **end while**

3.2. Dynamic Occupy Operator

Concerning re-entry path planning, omitting cells exist in two states. In order to monitor the state changes, a local dynamic Occupy operator is introduced in the paper. When the omitting cell (C) is blocked by dynamic obstacles, the Occupy operator has this cell blocked along the current path, and a new path is planned.

As indicated below, the Occupy operator is conducive to conducting local search in Algorithm 2. When its cleaning environment changes, the robot reacts promptly to identify similar cells in the adjacent areas of blocked cells and make them inter-connected. Formula (7) is the neighborhood of Cell C. In Formula (8), the information of changing cells is stored in the Darray. That is to say, input variable of Algorithm 2 is stored:

$$\{C_L, C_R\} \in U(C, \delta) \tag{7}$$

$$Darray = S_i - S' \tag{8}$$

Algorithm 2 Occupy ( )

Input: *Darray*
Output: S'
1: **for** each individual S_i in P **do**
2: find the two neighbour cities C_L and C_R of C in S_i
3: occupy C from S_i and link C_L and C_R to form new route S'
4: replace S_i by S'
5: **end for**

Figure 5 presents a simple instance that the Occupy operator solves problems of blocked cells. An assumption is made that the Cell D is occupied on the path and the Occupy operator will locally search adjacent cells of that cell to constitute a new path. To be specific, the path changes from

A-B-C-D-E-F to A-B-C-E-F. This operator is highly effective in solving algorithms practically, as the time for processing dynamic cells with this algorithm is fixed and the time complexity is O (1).

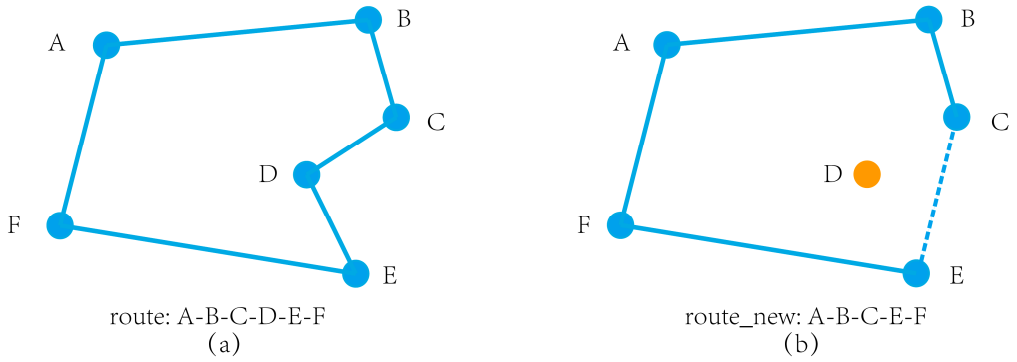

Figure 5. Practical instance of occupy operator: (**a**) static route; (**b**) new route by occupy operator.

3.3. Dynamic Inver-Over Evolutionary Algorithm

Following integration, the Occupy operator is capable to convert the static Inver-Over evolutionary algorithm for TSP into the dynamic Inver-Over evolutionary algorithm for the purpose of re-entry path planning.

As shown in Algorithm 3, non-cleaned cells shall be coded in pseudocode of the dynamic Inver-Over evolutionary algorithm, and the each population element is taken as the number of a cell. For instance, the gene sequence saved by the individual S_i containing information of 8 cells is 1-2-3-4-5-6-7-8, which indicates that the robot will depart from Cell 1 to pass by cells 2, 3, 4, 5, 6, 7 and 8 in succession. In this paper, path length is taken to evaluate whether individual genes are good. The individual genes are decomposed into a number of individual phenotypes. Then, individual path $length(S_i)$ is calculated, so:

$$length(S_i) = \sum_{j=1}^{n} \boldsymbol{d}(n_j, n_{j+1}) \tag{9}$$

$$\boldsymbol{d}(n_j, n_{j+1}) = \sqrt{(y_{j+1} - y_j)^2 + (x_{j+1} - x_j)^2} \tag{10}$$

where n indicates the number of cells, while $d(n_j, n_{j+1})$ denotes the distance between Cells j and $j + 1$. Individuals with higher $length(S_i)$ will be eliminated with algorithms.

Algorithm 3 Dynamic Inver-Over ()

Input: locations of cities and *Darray*
Output: dynamic optimal route
1: load the uncleaned cell locations and initialize the population P and $t = 0$
2: output the best route by Inver-Over($P(t)$)
3: **while** not arrive end time **do**
4: **while** *Darray* is not empty **do**
5: *cell_occupy* = *Darray*
6: Occupy(*cell_occupy*)
7: **end while**
8: output the best route in $P(t)$
9: **end while**

In Algorithm 3, Inver-Over operator plays significant roles. This operator primarily involves two search processes, which are mutation and crossover operations. Each individual performs an operation at probability k. After new solutions are determined, both types of operations are compared

against each other in terms of their strengths and weaknesses, in order to preserve the best solutions. For mutation operation, two cells are selected from a sequence on a random basis, and the content between two cells is completely reversed, as illustrated in Figure 6a. As for crossover operations, a cell is chosen from the current Individual 1, and a cell is selected from another Individual 2. Reversion will be made unnecessary if the two cells are adjacent. On the contrary, the content between the two cells will be completely reversed, as shown in Figure 6b. Both operations of the Inver-Over operator are conducive to an effective convergence of the algorithm and ensuring the search for the global optimal solution.

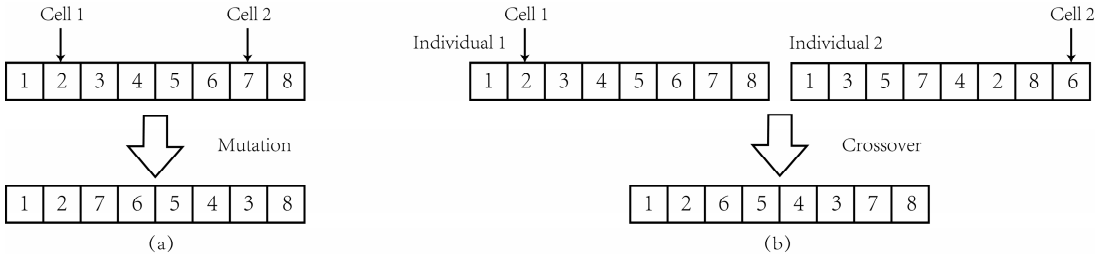

Figure 6. Execution process of Inver-Over operator: (**a**) Mutation; (**b**) Crossover.

Premised on the static Inver-Over evolutionary algorithm, a group of solutions appropriate for the identification of the optimal path can be obtained. Despite this, this path remains not suited to coping with environmental changes. Nevertheless, this problem is addressed by the dynamic Occupy operator in Algorithm 3. When the cleaning robot works under the backtracking mode, the cells are possibly occupied by dynamic obstacles. Under this circumstance, the overall number of cells will vary. To find out about which cells are blocked, the location information of block cells is saved during the robot's work with the assistance of the Darray.

Besides, the data in the Darray are further processed by the Occupy operator. This operator collects global information by making full use of static algorithm and avoids redundant operations for algorithm to re-start. For changing cells of environment, it is prompt to determine a new path by connecting and occupying adjacent cells. Therefore, it can be easily executed. While making response in real time, it reduces occupation of computer resources and is capable to be put into industrial practices with more ease. In combination with the kinematic model of robots introduced in Section 2.3, the corresponding commands are issued to the motor in the real world. Then the robot will move along the trajectory planned by the dynamic Inver-Over evolutionary algorithm.

In comparison to other evolutionary algorithms, the number of iterations is not taken as parameter in the dynamic algorithm. Instead, the actual system time is involved as parameter of this algorithm, largely because the environment rather than number of iterations changes over time in practice.

Dual independent hierarchical designs are introduced for the dynamic algorithm proposed in the paper. To be specific, they operate independently for global static search and local dynamic search. In case of any change to the state of a omitting cell, the dynamic Occupy operator will transform the old individual into a new one. The static search will be further optimized by the Inver-Over operator. If Darray remains void, the dynamic Inver-Over evolutionary algorithm will be degraded into static TSP algorithm. The effectiveness of the algorithm in solving re-entry path planning problems is reliant on the efficiency of the Inver-Over operator and high dynamic performance of the Occupy operator.

4. Experimental Simulations

4.1. Evaluation Indexes

The DTSP algorithm represents a solver for real-time approximate optimum path, which changes over time. To assess whether or not the dynamic Inver-Over evolutionary algorithm is effective, this paper reckons the errors between solutions of the dynamic algorithm and the static TSP algorithm as evaluation indexes.

The formulas of absolute error Δe and relative error $\bar{e}$ are expressed as follows:

$$\Delta e(t) = L(t) - L^*(t) \tag{11}$$

$$\bar{e}(t) = \frac{L(t) - L^*(t)}{L^*(t)} \tag{12}$$

where, $L^*(t)$ represents the optimal path length determined by the static TSP algorithm which is Inver-Over evolutionary algorithm. In this paper, the results obtained from the static TSP algorithm are hypothesized to be optimum. $L(t)$ indicates the optimum solution at t determined by the DTSP algorithm which is dynamic Inver-Over evolutionary algorithm proposed in this paper.

In the experimental process, all tests are performed for a space of 200 s and the optimal path is output every 1 s. Thus, the following results are obtained:

Maximum error:

$$\Delta e_{max} = \max_{t=1,\ldots,m} \{\Delta e(t)\} \qquad \bar{e}_{max} = \max_{t=1,\ldots,m} \{\bar{e}(t)\} \tag{13}$$

Minimum error:

$$\Delta e_{min} = \min_{t=1,\ldots,m} \{\Delta e(t)\} \qquad \bar{e}_{min} = \min_{t=1,\ldots,m} \{\bar{e}(t)\} \tag{14}$$

Mean error:

$$\Delta e_a = \frac{1}{m}\sum_{t=1}^{m} (\Delta e(t)) \qquad \bar{e}_a = \frac{1}{m}\sum_{t=1}^{m} (\bar{e}(t)) \tag{15}$$

4.2. Experimental Setup

In this paper, 14 nodes are created at random within a 20×20 square to simulate cells which are left uncleaned as cleaning robots avoid obstacles. To describe dynamic experimental environment, probability (p) and frequency (f) of environmental change are introduced in this paper. Where, p represents the probability of environmental change at t, and f indicates the ratio of the number of cells occupied at the time of environmental change to the overall number of cells.

When the probability of environmental change is relatively low, p value is lower, which suggests that environment is not dynamic. On the contrary, it is highly dynamic. The frequency of environmental change (f) reflects the extent of environmental change. In this paper, p and f are set to be 0.5 and 0.4 respectively. During experimental operations, f is randomly set to be positive between 0 and 0.4, for the purpose of realizing environmental changes at varying degrees under experimental environment. In doing so, a random dynamic work environment is created for cleaning robots. Under this circumstance, each path is free to change, and it is not directly associated with original environment at all.

Parameter selection for the static Inver-Over evolutionary algorithm could have a relatively significant impact on the final error analysis. Under such a small-scale TSP environment, this paper set the following parameters for static operations: total number of iterations = 1000; population size = 100; reversion probability (k) of Inver-Over operator = 0.4. To a certain degree, setting a higher number of iterations helps ensure that the static algorithm can determine the optimal path. The dynamic Inver-Over evolutionary algorithm is identical to the static evolutionary algorithm in respect of parameters. The distinction between these two algorithms lies in the fact that in the dynamic algorithm, system time is treated as end condition, while p and f are integrated into this algorithm.

All experimental parameters are indicated in Table 1 and their definitions are the same as those mentioned above. All the data involved in this paper were determined by programming through MATLAB R2016a using the Intel i5-6300HQ processor of the personal computer with a memory of 8 G.

Table 1. Experimental parameters.

Parameters	Value
iterations	1000
pop_size	100
k	0.4
p	0.5
f	0.4
m	200

4.3. Analysis on Experimental Results

Figure 7 shows work environment of a cleaning robot. The red nodes denote non-cleaned cells, which shall be traversed again by the robot. The two curves shown in Figure 8 present the optimal path length determined at the time of sampling by the dynamic Inver-Over evolutionary algorithm and the static algorithm respectively. As demonstrated in the figure, the results of both algorithms are the same at certain time points. At these time points, no cell is found occupied, with the dynamic algorithm degraded into the static algorithm.

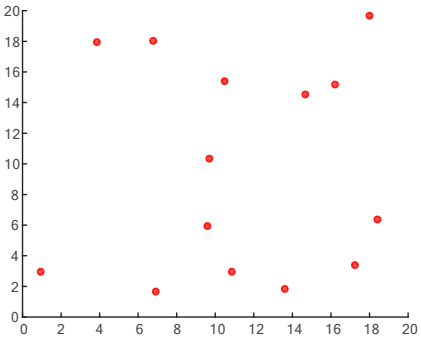

Figure 7. Non-cleaned cells.

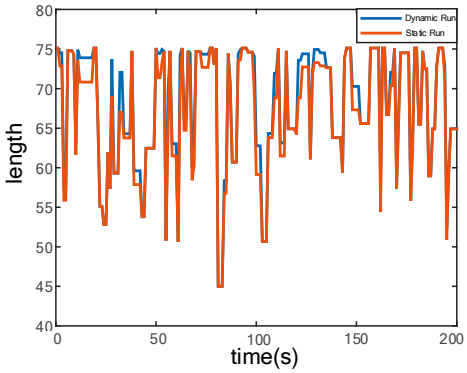

Figure 8. Curves on optimal path length determined by both algorithms.

Figure 9 presents the optimal paths of both algorithms at 1 s, 75 s, 150 s and 200 s, respectively. The red path represents the results obtained from the dynamic algorithm, while the blue path indicates the results obtained from the static algorithm. At the time point (t) of 75 s, three cells are found occupied by dynamic obstacles under the environment. In order to address this problem with the dynamic algorithm, the globally planned old path is transformed into a new one by the Occupy operator, while the static algorithm is to restart. Though the optimal solution can be obtained from the static algorithm, it is time-consuming as compared to the dynamic algorithm.

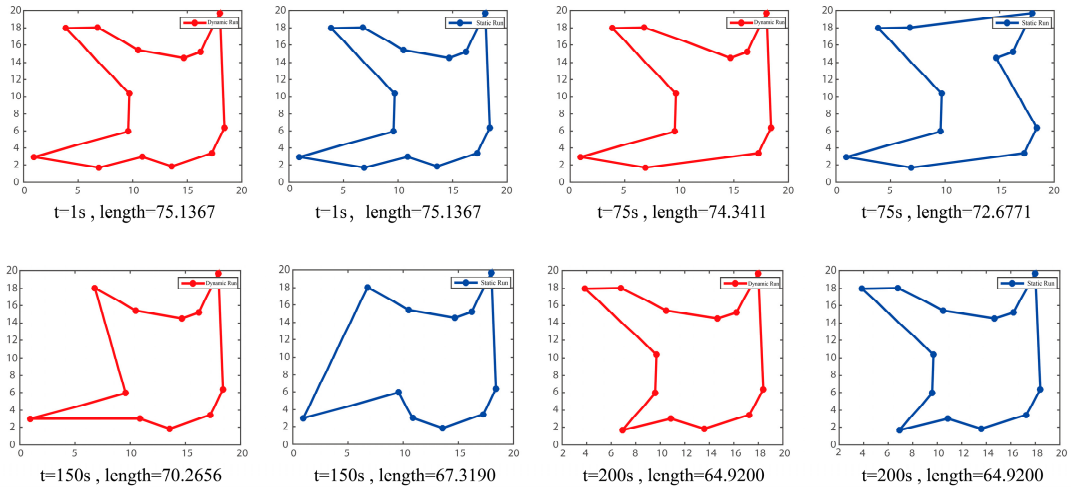

Figure 9. Optimal paths of both algorithms.

To illustrate the real-time performance of dynamic algorithm better, 10 sets of comparative experiments are performed with one cell occupied by the dynamic obstacle in the paper. The dynamic algorithm uses Occupy operator to calculate the results, while the static algorithm is to restart by inputting the cell position. The running time of both algorithms is shown in Figure 10. The results in Figure 10 indicate that the time required by the dynamic algorithm is far less than the static algorithm, proving the superiority of the dynamic algorithm in time.

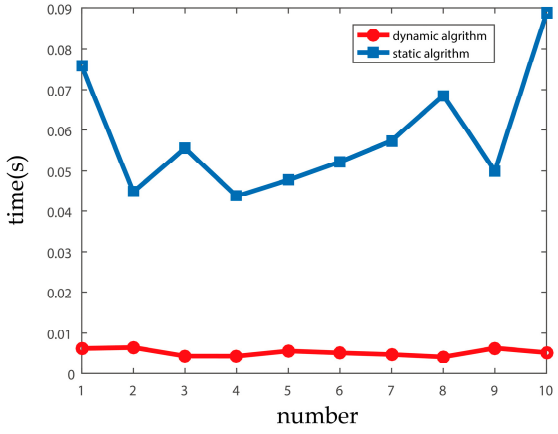

Figure 10. Running time of both algorithms.

In this paper, the results at the time of sampling were determined by the static Inver-Over algorithm, and compared against the results obtained from the dynamic algorithm to analyze errors. Figure 11 is the curve of relative errors, while Figure 12 is the curve of absolute errors. As shown in Figure 12, relative error is below 7% in most cases. More detailed results are indicated in Table 2, which indicates that the mean absolute error is 1.4574 and the mean relative error is merely 2.2%. Considering that the environment remains unchanged under some circumstances, the data without any error are discounted and deleted when the mean error is calculated. These results demonstrate that the dynamic Inver-Over evolutionary algorithm proposed in this paper is conducive to monitoring the best path under dynamic environment and making rapid responses. In addition, this algorithm is validated to be effective on addressing the problems with re-entry path planning.

Table 2. Errors between dynamic and static algorithms.

Error	Value
Δe_{max}	5.0552
Δe_{min}	0
Δe_a	1.4574
e_{max}	0.0754
$\bar{e}_{min}$	0
$\Delta \bar{e}_a$	0.022

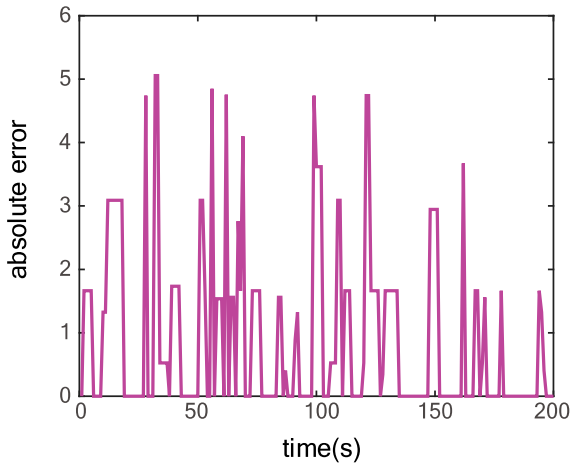

Figure 11. Curves of absolute errors.

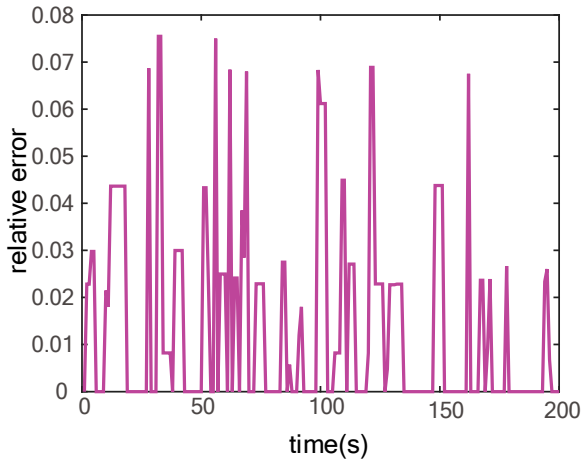

Figure 12. Curve of relative errors.

5. Conclusions

A path planning method for cleaning robots based on the dynamic Inver-Over evolutionary algorithm is proposed. A local dynamic operator is introduced and the static TSP algorithm is transformed into the dynamic Inver-Over evolutionary algorithm. As demonstrated by the simulation results, the path planning method is effective in tracking the optimal path under dynamic environment in real time. In the future, the actual energy consumption of robots will be taken into consideration. A complete technical scheme for planning robot paths shall be suggested in combination with the corresponding complete coverage path planning.

Author Contributions: Conceptualization, Y.T. and C.C.; Methodology, Y.T.; software, C.C.; writing—Original draft preparation, C.C.; writing—Review and editing, Y.T. and C.C. and T.W. and Y.C.; supervision, H.X. and F.R. and Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Beijing Advanced Innovation Center for Biomedical Engineering.

Acknowledgments: The authors would like to thank all the colleagues who contributed to this research work.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, T.M.; Tao, Y.; Liu, H. Current researches and future development trend of intelligent robot: A review. *Int. J. Autom. Comput.* **2018**, *15*, 525–546. [CrossRef]
2. Nakamura, K.; Nakazawa, H.; Ogawa, J.; Naruse, K. Robot sweep path planning with weak field constrains under large motion disturbance. *Artif. Life Robot.* **2018**, *23*, 532–539. [CrossRef]
3. Choset, H. Coverage for robotics–A survey of recent results. *Ann. Math. Artif. Intell.* **2001**, *31*, 113–126. [CrossRef]
4. Choset, H.; Pignon, P. Coverage path planning: The boustrophedon cellular decomposition. In *Field and Service Robotics*; Springer: London, UK, 1998; pp. 203–209.
5. Wang, J.Y.; Chen, J.; Cheng, S.; Xie, Y. Double heuristic optimization based on hierarchical partitioning for coverage path planning of robot mowers. In Proceedings of the 2016 12th International Conference on Computational Intelligence and Security (CIS), Wuxi, China, 6–19 December 2016; pp. 186–189.
6. Pitsch, M.; Pryor, M. Temporally static environment coverage with offline planning techniques. In Proceedings of the 2017 IEEE Workshop on Advanced Robotics and Its Social Impacts (ARSO), Austin, TX, USA, 8–10 March 2017; pp. 1–2.
7. Miao, X.; Lee, J.; Kang, B.Y. Scalable Coverage Path Planning for Cleaning Robots Using Rectangular Map Decomposition on Large Environments. *IEEE Access* **2018**, *6*, 38200–38215. [CrossRef]
8. Viet, H.H.; Dang, V.H.; Laskar, M.N. BA*: An online complete coverage algorithm for cleaning robots. *Appl. Intell.* **2013**, *39*, 217–235. [CrossRef]
9. Ma, Y.F.; Sun, H.; Ye, P.; Li, C. Mobile robot multi-resolution full coverage path planning algorithm. In Proceedings of the 2018 5th International Conference on Systems and Informatics (ICSAI), Nanjing, China, 10–12 November 2018; pp. 120–125.
10. Gabriely, Y.; Rimon, E. Spanning-tree based coverage of continuous areas by a mobile robot. *Ann. Math. Artif. Intell.* **2001**, *31*, 77–98. [CrossRef]
11. Khan, A.; Noreen, I.; Habib, Z. On Complete Coverage Path Planning Algorithms for Non-holonomic Mobile Robots: Survey and Challenges. *J. Inf. Sci. Eng.* **2017**, *33*, 101–121.
12. Yang, S.X.; Luo, C. A neural network approach to complete coverage path planning. *IEEE Trans. Syst.* **2004**, *34*, 718–724. [CrossRef] [PubMed]
13. Luo, C.; Yang, S.X. A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments. *IEEE Trans. Neural Netw.* **2008**, *19*, 1279–1298. [CrossRef]
14. Yakoubi, M.A.; Laskri, M.T. The path planning of cleaner robot for coverage region using genetic algorithms. *J. Innov. Digit. Ecosyst.* **2016**, *3*, 37–43. [CrossRef]
15. Schafle, T.R.; Mohamed, S.; Uchiyama, N. Coverage path planning for mobile robots using genetic algorithm with energy optimization. In Proceedings of the 2016 International Electronics Symposium (IES), Denpasar, Indonesia, 29–30 September 2016; pp. 99–104.
16. Pham, H.V.; Lam, T.N. A new method using knowledge reasoning techniques for improving robot performance in coverage path planning. *Int. J. Comput. Appl. Technol.* **2019**, *60*, 57–64. [CrossRef]
17. Zhou, Y.; Sun, R.; Yu, S. A Complete Coverage Path Planning Algorithm for Cleaning Robots Based on the Distance Transform Algorithm and the Rolling Window Approach in Dynamic Environments. In Proceedings of the 2017 IEEE 7th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Honolulu, HI, USA, 31 July–4 August 2017; pp. 1335–1340.
18. Qiu, X.; Song, J.; Zhang, X.; Liu, S. A complete coverage path planning method for mobile robot in uncertain environments. In Proceedings of the 2006 6th World Congress on Intelligent Control and Automation, Dalian, China, 21–23 June 2006; pp. 8892–8896.

19. Yang, J.; Ding, R.; Zhang, Y. An improved ant colony optimization (I-ACO) method for the quasi travelling salesman problem (Quasi-TSP). *Int. J. Geogr. Inf. Sci.* **2015**, *29*, 1534–1551. [CrossRef]
20. Deng, Y.; Liu, Y.; Zhou, D. An improved genetic algorithm with initial population strategy for symmetric TSP. *Math. Probl. Eng.* **2015**. [CrossRef]
21. Bello, I.; Pham, H.; Le, Q.V. Neural combinatorial optimization with reinforcement learning. *arXiv* **2016**, arXiv:1611.09940.
22. Mahi, M.; Baykan, Ö.K.; Kodaz, H. A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem. *Appl. Soft Comput.* **2015**, *30*, 484–490. [CrossRef]
23. Tao, G.; Michalewicz, Z. Inver-over operator for the TSP. In Proceedings of the International Conference on Parallel Problem Solving from Nature, Amsterdam, The Netherlands, 27–30 September 1998; Springer: Berlin/Heidelberg, Germany, 1998; pp. 803–812.
24. Yan, X.; Liu, H.; Yan, J.; Wu, Q. A fast evolutionary algorithm for traveling salesman problem. In Proceedings of the Third International Conference on Natural Computation (ICNC 2007), Haikou, China, 24–27 August 2007; pp. 85–90.
25. Arshad, S.; Yang, S. A hybrid genetic algorithm and inver over approach for the travelling salesman problem. In Proceedings of the IEEE Congress on Evolutionary Computation, Barcelona, Spain, 18–23 July 2010; pp. 1–8.
26. Wang, Y.; Sun, J.; Li, J.; Gao, K. A modified inver-over operator for the traveling salesman problem. In Proceedings of the International Conference on Intelligent Computing, Zhengzhou, China, 11–14 August 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 17–23.
27. Singh, D.R.; Singh, M.K.; Singh, T. A Hybrid Algorithm with Modified Inver-Over Operator and Genetic Algorithm Search for Traveling Salesman Problem. In Proceedings of the Advanced Computing and Communication Technologies, Panipat, India, 18–20 November 2016; Springer: Singapore, 2016; pp. 141–150.
28. Huang, Z.C.; Hu, X.L.; Chen, S.D. Dynamic traveling salesman problem based on evolutionary computation. In Proceedings of the 2001 Congress on Evolutionary Computation, Seoul, Korea, 27–30 May 2001; pp. 1283–1288.
29. Zhou, A.; Kang, L.; Yan, Z. Solving dynamic TSP with evolutionary approach in real time. In Proceedings of the 2003 Congress on Evolutionary Computation, Canberra, Australia, 8–12 December 2003; pp. 951–957.
30. Li, C.; Yang, M.; Kang, L. A new approach to solving dynamic traveling salesman problems. In Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning, Hefei, China, 15–18 October 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 236–243.