electronics

MDPI

Article

A Hybrid Tabu Search and 2-opt Path Programming for Mission Route Planning of Multiple Robots under Range Limitations

Meng-Tse Lee [1,*], Bo-Yu Chen [1] and Ying-Chih Lai [2,*]

1 Department of Automation Engineering, National Formosa University, Yunlin 632, Taiwan; 40127227@gm.nfu.edu.tw
2 Department of Aeronautics and Astronautics Engineering, National Cheng-Kung University, Tainan 701, Taiwan
* Correspondence: mtlee@nfu.edu.tw (M.-T.L.); yingclai@mail.ncku.edu.tw (Y.-C.L.); Tel.: +886-5-631-5388 (M.-T.L.); +886-6-2757575 (ext. 63648) (Y.-C.L.)

Received: 28 February 2020; Accepted: 23 March 2020; Published: 24 March 2020

Abstract: The application of an unmanned vehicle system allows for accelerating the performance of various tasks. Due to limited capacities, such as battery power, it is almost impossible for a single unmanned vehicle to complete a large-scale mission area. An unmanned vehicle swarm has the potential to distribute tasks and coordinate the operations of many robots/drones with very little operator intervention. Therefore, multiple unmanned vehicles are required to execute a set of well-planned mission routes, in order to minimize time and energy consumption. A two-phase heuristic algorithm was used to pursue this goal. In the first phase, a tabu search and the 2-opt node exchange method were used to generate a single optimal path for all target nodes; the solution was then split into multiple clusters according to vehicle numbers as an initial solution for each. In the second phase, a tabu algorithm combined with a 2-opt path exchange was used to further improve the in-route and cross-route solutions for each route. This diversification strategy allowed for approaching the global optimal solution, rather than a regional one with less CPU time. After these algorithms were coded, a group of three robot cars was used to validate this hybrid path programming algorithm.

Keywords: multi-robots; path programming; tabu search

1. Introduction

Mainstream applications are currently focused on unmanned vehicle robots used in manufacturing; unmanned air vehicles (UAVs) in monitoring the earth's surface; emergency aid and disaster control and prevention efforts; commercial aerial photography; logistics; and unmanned combat air vehicle operations (UCAVs) [1,2]. When the scope of tasks and the areas involved are expanding, a system consisting of multiple unmanned vehicles agents is required to complete a mission with a very wide area. To complete the tasks more efficiently, well-planned path programming is a must, so as to minimize time and energy consumption by shortening the overall distances of the routes.

Facing this problem of a large area multi-waypoints mission dealing with a multi-agent system, we designed a hybrid dynamic path programming algorithm to help us achieve the goals of saving time and energy, with shorter and more efficient routes so that the robot cars were not running redundant paths. The maximum travelable distance (limited by the battery energy capacity) was used as one of the constraints during the algorithm iterations.

Unlike other path-programming works, this study contributes to the field with a hybrid path programming algorithm involving a combined tabu search and a 2-opt swap under the maximum travelable range consideration. A set of multiple robot cars was built to validate the tasks' completeness in the final phase.

2. Related Work

The idea of using a multi-agent robot system has been gradually adapted for wide-area searching in large-scale missions. This includes the use of multi-robot Simultaneous Localization and Mapping (SLAM), as explained by Atanasov et al. [3], along with hitchhiking robots as part of a collaborative approach for efficient multi-robot navigation, as explored by Ravankar et al. [4]. However, neither of these studies were focused on including an optimized mission path for each robot of the system. In fact, many scholars have been conducting research on path programming, as well as various factors regarding the number of vehicles dispatched. Most research is focused on solving the Vehicle Routing Problem (VRP) and the multiple traveling salesman problem (mTSP). This mTSP involves m salesmen who must visit a set of n cities, with each salesman starting and ending at the same place (city). Each city must be visited exactly once by one salesman, with the objective of finding the shortest total distance traveled by all of the salesmen. It could happen that we have one of the salesmen travel to all of the cities, while others visit only one. However, in practice, every salesman has similar abilities and limits. So the mTSP with ability constraints is more appropriate for real-world problems. Suppose the number of cities traveled to by each salesman is limited [5]. The multi-robot, multi-target exploration problem further extends the traveling salesman problem (TSP). This problem is called the Multiple Traveling Robot Problem (MTRP), and it involves a team of robots visiting target points at least once (ideally, no more than once). The overall solution quality is dependent upon both the quality of the solution constructed by the paths of the robots, and the efficient allocation of the targets to those robots [6].

The mTSP is a generalization of the well-known TSP [7], which is an obviously non-deterministic polynomial-time hardness (NP-hard) problem [8,9]. With NP-hard problems, the complexity of the solution increases as the number of target points expands. Since it is almost impossible to go through all of the feasible paths, and produce the best solution within an acceptable timeline, a heuristic algorithm was adopted to obtain an approximate solution.

Dorigo et al. [10] suggested a new computational paradigm called ant system (AS) for solving the TSP problem. This innovative contribution formed the foundation of today's Ant Colony Optimization (ACO) algorithm.

Yousefikhoshbakht et al. [11] used the New Modified Ant Colony Optimization (NMACO) that mixed the insert, swap and 2-opt algorithms, as well as an efficient Candidate List to improve the efficiency of the ACO. It is quite competitive with other meta-heuristic algorithms that solve a library of sample instances for the TSP (TSPLIB) instances. But the solution it yielded still has room to improve.

Necula et al. [12] used five modified Ant Colony Optimization (ACO) algorithms to solve mTSP. With this method, the maximum number l and the minimum number k of the city that each salesman should visit are defined and limited. The mTSP is then solved with TSPLIB instances.

Xu et al. [13] used a hybrid Genetic Algorithm (GA) and Simulated Annealing (SA) to solve mTSP. This enhanced the local search ability so as to achieve a better global optimal solution than general GA. But the solution it solved is not the best. In the previously mentioned literature, the authors used ACO and GA to solve the mTSP problem, but it does not produce the same solution with every execution, and must be tested continuously to ensure the best solution.

Côté et al. [14] used Tabu Search (TS) to solve the VRP, which they tested with benchmark instances. They not only achieved convergence in a reasonable time, but also surpassed many of the best solutions in benchmark instances.

Huang and Liao [15] used a two-phased method to solve the Dynamic Vehicle Routing Problem (DVRP). Different from mTSP, it limits its vehicle capacity. Fuzzy c-mean techniques were used to divide existing customers, while cost function was adopted to figure out the initial solution in the first phase. Then, in the second phase, a tabu search was combined with 2-opt and or-opt to improve cross-route and in-route respectively, aiming to solve the problem of multi-vehicle paths programming.

In fact, the tabu search algorithm not only adapted in path programming, but also in many other fields, such as those involving the economic dispatch of electric generators [16].

Sariel-Talay et al. [6] studied MTRP. They mentioned that their system was able to obtain real-time, optimal paths for traveling among multiple target points on their own platform with a robot swarm, and the assignment was completed successfully. However, a single robot car's maximum traveling capacity was not considered.

There are many published articles on multi-vehicle path programming, but only a few involve on-site experiments with cars to verify the effectiveness of the programming. In mTSP, the maximum and minimum number of cities that each salesman should visit are defined and limited, but the tasks may not get completed due to the limited energy capacity of one car. Therefore, we focused on the "maximum travelable distance" for this study, and also adopted a two-phased module as our solving module. In the first phase, a tabu search combined with a 2-opt swap method was used to program a single optimal path, and then to get the initial solution by splitting the path into multiple sub-paths. In the second phase, Huang and Liao's solving module was adopted—a tabu search combined with a 2-opt swap method to improve the initial paths, and to determine whether the result exceeded the maximum distance limit.

Additionally, a diversification strategy similar to GA was implemented for in-route path improvements. The use of this strategy resulted in the recording and selection of the optimal solution or second-best solution as the initial solution for the in-route path improvement later on, so that an unwanted regional optimal solution could be avoided. In this research, the 2-opt swap method was adopted to improve in-route paths. Once it finished, the calculation continued to improve cross-route paths until the conditions of termination were reached. In the last stage, this solution was verified by our experiments with real robot cars running the programmed paths. From the experiment, we looked into the distance designed for the path programming, and the trajectory difference between the actual paths and theoretical ones.

Research Highlights

The objective of the path programming problem of this research was to assign several series of target points to multi-robots to maximally reduce total energy consumption. The robots had limited onboard (fuel or battery) energy, so a maximum travelable distance constraint was expected to be satisfied. Highlights of this research speak to these issues:

1. We developed an optimal route planning algorithm for multi-robots' application by using an innovative, hybrid, two-phase tabu and 2-opt search.
2. In previous research involving salesman problems, constraints in VRP, mTSP and MTRP were mainly located on maximum reachable target points for single vehicle problems, or on maximum cargo capacity for multiple vehicles problems. In our study, we made this problem closer to an unmanned system's reality by building a new model with maximum range constraint.
3. Unlike most previous research in VRP and mTSP, in which problems end with a computational result, outdoor field tests were conducted to validate the algorithm developed in this project.

3. Design of the Two-Phase Path Programming

This research was focused on the dynamic path programming of multi-robots for achieving missions that lowered the energy costs in each group. The problem defined in this research is similar to the mTSP problems of visiting n targets with a shortest total route-distance by using m vehicles, with every target being visited only once.

However, the service range of unmanned robot vehicle systems is strictly limited by its onboard energy capacity (such as fuel or battery); hence the major difference in this research, as compared to traditional mTSP projects, is consideration of the range-limitation as a constraint during the programming loops. According to the previous statement, the problem definition for this research was modified from a standard module of SD-mTSP [17]. The object and subject were as follows:

$$x_{ijk} = \begin{cases} 1 & \text{the k path goes from city i to city j} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$\forall(\mathrm{i,j}) \in \mathrm{A},\ k = 1,\dots m \tag{2}$$

Object:

$$min \sum_{k=1}^{m} \sum_{(i,j)\in A} c_{ij}x_{ijk} \tag{3}$$

Subject to:

$$\sum_{(i,j)\in A} c_{ij}x_{ijk} \le D_{lmt}, k = 1,\dots m \tag{4}$$

$$\sum_{k=1}^{m} \sum_{j=2}^{n} x_{1jk} = m \tag{5}$$

$$\sum_{k=1}^{m} \sum_{j=2}^{n} x_{j1k} = m \tag{6}$$

$$\sum_{k=1}^{m} \sum_{i=1}^{n} x_{ijk} = 1\ ,\ j = 2,\ \dots, n \tag{7}$$

$$\sum_{k=1}^{m} \sum_{j=1}^{n} x_{ijk} = 1\ ,\ i = 2,\ \dots, n \tag{8}$$

$$\sum_{k=1}^{m} x_{i1k} + x_{1ik} \le 1\ ,\ i = 2,\ \dots, n \tag{9}$$

$$u_i \ge 1,\ i = 2,\ \dots, n \tag{10}$$

where c_{ij} is the distance array of A; m is the number of robot cars; D_{lmt} is the value of maximum distance limit; n is the number of target points; and u_i is the number of target points visited on a car's path from the original point to target I. Equation (1) is variable integer, determining whether target i to j has been traveled by k car; with Equation (2), A in the formula is the set of all specified paths; Equation (3) is the object function of this problem; in Equation (4) the value of subject D_{lmt} should not be exceeded by all of the cars; Equations (5) and (6) ensure that cars start at the original point and come back to the same point; Equations (7) and (8) ensure that all of the targets have been entered and exited by the car; Equation (9) indicates that each car must visit at least one target; and Equation (10) prevents the problem of the sub-path not including the original point.

There are many kinds of heuristic algorithms to solve this problem, including ACO, Particle Swarm Optimization (PSO), GA and tabu search. In this research we mainly adopted the tabu search of a heuristic algorithm that can imitate human beings' memory, which keeps experiences in the past to prevent roundabout searches and to create a tabu list. It can learn from past solutions to avoid any regional optimal solution being seen as a global optimal solution. Tabu search is known for its ability to quickly converge iterations. In addition, its optimal solution and the number of iterations of convergence are more stable than what GA can achieve.

As GA randomly generates paths by mating, the solution could have turned out to be a suboptimal rather than an optimal solution, so we needed to repeatedly verify the result for the ultimate optimal solution. We knew that as long as the initial solution and tabu list for tabu search were well set, we could quickly complete a convergence and get the more stable, optimal solution; that being the main reason we adopted it for this study [2].

With tabu search as the core technology and involving the 2-opt swap method, a two-phase path programming algorithm model was established for this research. As shown in Figure 1, the initial solution established in the first phase was processed in two steps.

In the second phase, we needed to obtain the global optimal solution by improving the initial one we realized in the first phase. As in the first phase, three steps were used. Our first was the cross-route improvement by tabu search combined with 2-opt to exchange different target points within different paths. Then, a diversification strategy that recorded the optimal and suboptimal solutions from last computation was implemented to avoid the result falling into a regional optional solution. Step 3 was to improve each in-route path with tabu search combined with 2-opt. In other words, it was a refined solution exchanged from the first step, and we needed to repeat the process we carried out in step 2 until a termination condition was reached.

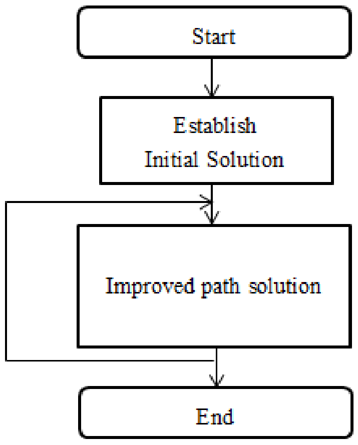

Figure 1. Main flow chart.

3.1. *Tabu Search*

Tabu search is a global search method. First, it establishes an initial solution, and then finds the neighborhood optimal solution, or accords the solution of aspiration criterion as the base for moving. That means, searching for solutions in the neighborhood domain of the current solution. Among them, the tabu list memory mechanism is noticeably important. It records the solutions which have been searched already to prevent any useless or redundant searching. Once the search on all neighborhood domains is completed, the optimal solution is selected. If any solution that be selected is found to be better than the current optimal solution, the optimal solution is updated until the termination condition is reached [18].

3.2. *2-opt Swap Method*

The tabu search combined with the 2-opt nodal line swap method that was proposed by Lin (1965) [19] was adopted in this research. It is a method that we used to change the order of the path to expand the current solution. Initially it was designed on TSP, and now it is widely used in solving path problems (TSP, VRP, VRPTW, and so on). Its swap concept is shown in Figure 2. If (1, 3) and (2, 4) nodal lines are replaced, (1, 2) and (3, 4) could be connected to change its path.

The method for the cross-route path swap is different from the one we used for the single one path (see Figure 3 for the swap concept). If the nodal lines of (5, 6) and (1, 2) are exchanged, (5, 2) and (1, 6) as well as (5, 1) and (2, 6) are two possible paths that will be changed, respectively. Compared to the original path, the direction of the latter one will change. Thus, the use of 2-opt would be possible for reversing the directions of the paths.

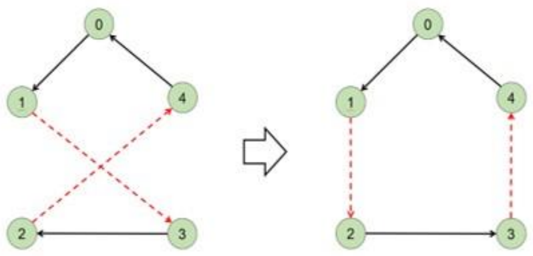

Figure 2. 2-opt Swap Method Concept (a).

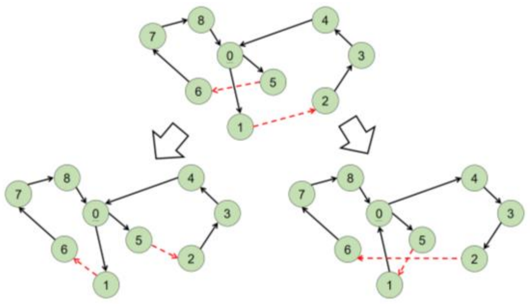

Figure 3. 2-opt Swap Method Concept (b).

3.3. *Establishment of the Initial Solution*

This phase was carried out in three steps, as shown in Figure 4. In step 1, tabu search and 2-opt were used to optimize the single path. In step 2, the single path was being divided into multiple sub-paths as the initial solution for our second phase.

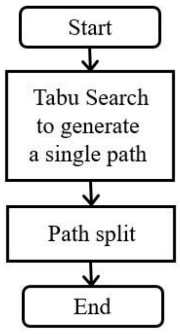

Figure 4. Establishment of the Initial Solution.

3.3.1. Use of Tabu Search to Program a Single Path

We used Nearest Distance Method to establish a rough single path as an initial solution for the tabu search. First, we established a distance array c_{ij}, and let the zeroth row of $c_{ij} = \infty$. Next, we set the point in which the row of c_{ij} had a minimum value as the first point, and let the value of this row of c_{ij} be infinity. Then, we repeated the same procedure to search and set the second point, the third point, and so on. The initial solution was generated until all of the points were searched. Following this, the tabu search and 2-opt were used to optimize the single path. The 2-opt was used as a move method to solve the problem of the single optimal path. We also set up the length of the tabu list, with the condition of stopping the search and using the solution of the Nearest Distance Method as the initial solution x_0 at that time. We then executed a tabu search to get the optimal single path.

3.3.2. Path Split

We roughly split the single path program into multiple sub-paths, and then split all the target points into m sub-paths. We also made the closed sub-paths link to the original point as much as possible, less than the maximum distance limit. At worst, no more than one route exceeded the maximum distance limit. The split path was not necessarily the optimal path, but it was good enough to be the initial solution for the next phase.

3.4. Improvement of Path to Obtain an Optimal Solution

In this phase, the rules of 2-opt were followed to swap the node lines in in-route and cross-route paths for moving. The solutions were gradually converged into the optimal solution. At this point, we decided to minimize the distance summation of all paths, and make sure the assigned distance of each robot car did not exceed the maximum distance limit. The flow chart is shown in Figure 5.

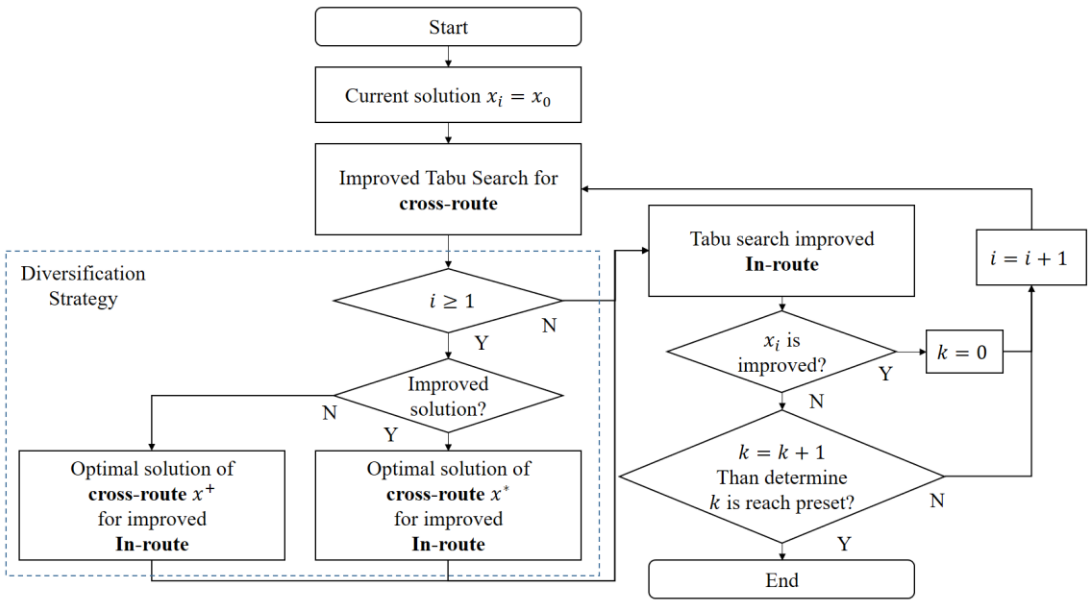
Start
Current solution $x_i = x_0$
Improved Tabu Search for **cross-route**
Diversification Strategy
$i \geq 1$
N
Y
Improved solution?
N
Y
Optimal solution of **cross-route** x^+ for improved **In-route**
Optimal solution of **cross-route** x^* for improved **In-route**
Tabu search improved **In-route**
x_i is improved?
Y
$k = 0$
$i = i + 1$
N
$k = k + 1$ Than determine k is reach preset?
N
Y
End

Figure 5. Path Improvement Flow Chart.

3.4.1. Modified Tabu Search for Improving Cross-Route Switch

Compared to a general tabu search, the major difference of this improved search is that S_{lmt}, which contains solutions exceeding the maximum distance limitation, were removed from the Candidate List to ensure that all path solution values were within range. However, this limitation resulted in an empty set for the Candidate List, thereby blocking all solutions from entering the next generation. When this occurred, an original tabu list was adopted with S_{lmt}, the solution set retaining all solutions, even with values exceeding the maximum distance limitation. With loosened criteria, the search also selected "the distance of the longest traveling path" as the solution of minimum value.

A tabu search combined with 2-opt was used in this step. By moving based on 2-opt, swapping the nodal lines among different paths was used to improve each of the current paths. So, 2-opt was applied to improve cross-route paths. At that time, we set up the length of the tabu list, the condition of stopping the search, and the initial solution as x_0. Then we executed the flow of the improved tabu search to get the optimal paths as follows (its pseudo code is shown in Algorithm 1):

Step 1.

First, an initial solution x_0 was set as the current optimal solution; the suboptimal solution was x^*, x^+. Then we set the current optimal and suboptimal distance function values as $d(x^*), d(x^+)$; with the initial generation of $i = 0$, the number of optimal solutions was not improved $(k = 0)$ and tabu list T was the empty set $(T = \varnothing)$.

Step 2.

According to characteristics of 2-opt, it extends all feasible neighborhood solutions of x_i as a neighborhood solution set $N(x_i)$. The candidate list included an $N(x_i)$ deduction T, and a solution of the maximum distance limit S_{lmt} $[N(x_i) - T - S_{lmt}]$. We selected the optimal neighborhood solution x_{i+1} which had an optimal total distance function value of $d(x_{i+1})$ in it. $(x_{i+1} = MinTotalDist(N(x_i) - T - S_{lmt}))$. If $N(x_i) - T - S_{lmt} = \varnothing$, we changed the Candidate List to $N(x_i) - T$ and selected an optimal neighborhood solution x_{i+1}, which had an optimal maximum distance function value of $t(x_{i+1})$ in it. (Note that the $x_{i+1} = MinMaxDist(N(x_i) - T)$)

Step 3.

We had to determine whether $d(x_{i+1})$ was less than $d(x^*)$. If it was, x^* had to be replaced by x_{i+1} $(x^* = x_i)$. Then k = 0. Otherwise, the current optimal solution was preserved, and k = k + 1. Then, we determined whether $d(x_{i+1})$ was less than $d(x^+)$. If it was, x^+ was replaced with x_{i+1} $(x^+ = x_i)$ Otherwise, we proceeded to the next step.

Step 4.

We determined whether the condition of stopping the search was reached (stopping the search and setting the current optimal solution as the global optimal solution when *k* reached the preset). Otherwise, we updated the tabu list. That is, we added x_{i+1} to T. If T was full, according to rule of first-in, first-out, to evict the solution which is the earliest one that enters T. Then we let *i* = *i* + 1 and returned to Step 2 to continue the operation.

Algorithm 1. Improved Tabu Search Pseudo Code

```
Begin
$i, k = 0$
                         $x_i,\ x^* = x_0$
$T = \emptyset$
Do
Use 2-opt method to expand $x_i$ to get $N(x_i)$
Candidate List = $N(x_i) - T - S_{lmt}$
$x_{i+1}$ = Neighborhood optimum have minimum total distance in Candidate List
If Candidate List $= \varnothing$ then
                    Candidate List $= N(x_i) - T$
$x_{i+1}$ = Neighborhood optimum have minimum maximum-distance in Candidate List
end If
If $x_{i+1} < x^*$ then
$x^* = x_{i+1}$
$k = 0$
else
k = k + 1
If $x_{i+1} < x^+$ then
$x^+ = x_{i+1}$
end If
end If
add $x_{i+1}$ into $T$
$i = i + 1$
   While $k <=$ preset
end While
end
```

3.4.2. Diversification Strategy

When this phase proceeded to the second generation, part of the cross-route improvements had solutions that were not improved. It was possible for a solution to be identical to the one obtained in the first generation. If this solution was sent to the calculation in the next step, it still came out as a useless local optimal solution without any further improvements. In order to avoid this situation, we had to determine whether the solution was improved before going to the second generation. If not, the suboptimal solution from the cross-route path exchange had to be implemented for improvement, to get rid of the circle of the local optimal solution and to obtain the global optimal solution.

3.4.3. Improvements for Each Single Path

In this part, the path solution given by the diversification strategy was adopted as the initial solution. We improved the single path of each group by using a tabu search in combination with 2-opt to be the same as in the first stage. Thus, we used each single path as an initial solution, and we used 2-opt as a move method to make improvements among different paths. Additionally, we set up the length of the tabu list with the condition of stop search. Then we executed the tabu search that was the same as that of the first stage. We did this until each group had been improved after the end of the sequence.

4. Experimental Vehicle and System Architecture

4.1. System Architecture

As we aimed to complete the task with multiple target points by multiple vehicles in the shortest possible time, a multi-agent system (MAS) under central control was set up for this research. The so-called "central control" was enlisted to allocate assignments to each individual robot, so that each of them could complete the tasks independently, rather than controlling the cars throughout the procedure. Through cooperation among multiple agents, the system was able to complete a task of a larger scale. From an MAS viewpoint on task allocation, the system was seen as having a Single-robot Task, Single-task Robots, and an Instantaneous Allocation of Task (ST-SR-IA) [20].

Various assignments at different levels of difficulty were randomly allocated by the system, meaning that each robot car may have received any assignment. Thus, each car in the robot swarm was equipped with identical specifications to ensure their performance and capacity to cope with various assignments.

From the viewpoint of MAS heterogeneity, the degree of similarity among individual robots within a collection Het(R) can be expressed as follows:

$$\mathrm{Het(R)} = -\sum_{i=1}^{Caste} p_i \log_2(p_i) \tag{11}$$

where *Caste* represents types of robots, and p_i is the decimal percent of robots belonging to any caste. Since all specifications of this system are the same (caste = 1, $\mathrm{p_i}$ = 1), Het(R) = 0 (Equation (11)) [21], which indicates a homogeneity system of a robot swarm.

Figure 6 illustrates the system architecture. This system included a Remote Monitoring Station, a Multi-Vehicle Paths Programming System, and Robot Cars. The Remote Monitoring Station acted as a central controller capable of programming the paths and distributing paths to the robot swarm, designed to complete the work together. From the viewpoint of network topology, the system was a sort of star topology. Each robot car was able to communicate with the Remote Monitoring Station, but they were not able to communicate with each other. The system mainly used LabVIEW to develop the Dynamic Remote Monitoring Station interface. By sending a URL to obtain a Google map web page as a display interface, it was used as a man–machine interface (MMI), in which each vehicle's location feedback and initial path programming could be displayed and managed. By clicking the target points displayed on the MMI, the location could be sent to the Remote Monitoring Station, where the algorithm was run for the path programming. Then, the planned paths were passed to each car via XBee. Once the cars received the data, they cross-compared their locations against the target points designated by the central monitoring station. Next they came out at an angle between the car and target point, and then headed out of the electronic compass. With these results, the robot car was able to drive the DC motor forward and control the servo motor differential to successfully complete the assignment.

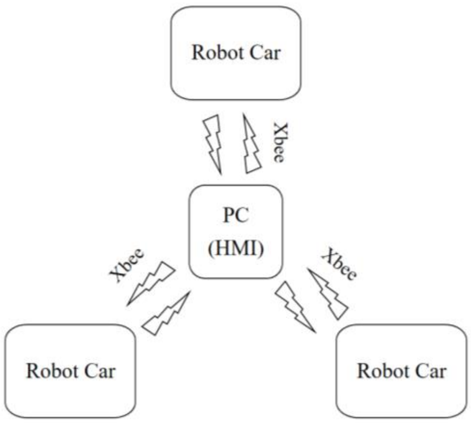

Figure 6. The Robot Cars System Architecture.

4.2. Experimental Vehicle

The robot car was used as an experimental vehicle (see Figure 7). Its function was to receive the data of target points from the Remote Monitoring Station and to establish a database for traveling. Once the car arrived at a designated target point, its real-time location was immediately sent to the Remote Monitoring Station.

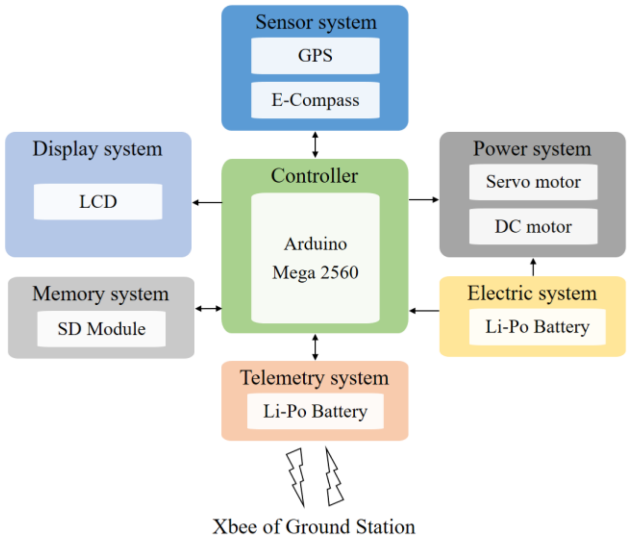

Figure 7. The Sub-System of Robot Car Architecture Diagram.

This experimental vehicle was modified from a 1/10 scale Shot Course Truck remote control car. We removed the car's shell and related remote control devices, then mounted additional off-the-shelf electronic components, including an Arduino Mega 2560 to act as an onboard computer; a U-blox NEO-7M Global Positioning System (GPS) module to provide position information for navigation; an HMC5883L Electronic Compass (E-Compass) to indicate heading; a 915 Mhz Xbee PRO S3B wireless module for duplex communication with the Remote Monitoring Station; a Liquid Crystal Display (LCD) to act as a health information indicator for field debugging; and a memory card (SD Module) to function as an integral data logger to include trajectory history. All the embedded automotive electronics are shown in Figure 8.

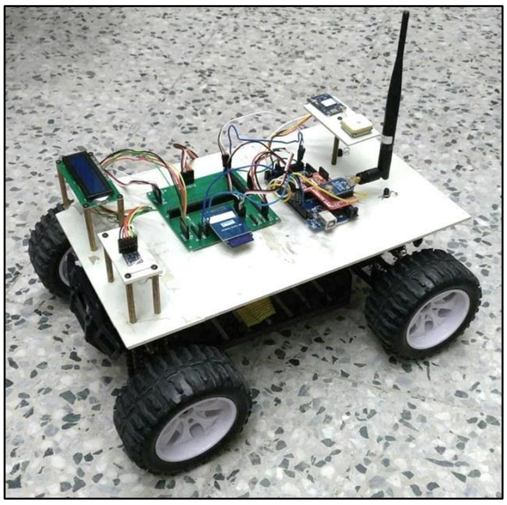
Figure 8. Experimental Robot Vehicle.

The robot car control principle was constructed from the latitude and longitude data of each target point stored in the database. Once the tasks assigned by the Remote Monitoring station were received by the robot car, the geographical data of each target point was extracted from the database for cars to travel among the points. On the other hand, a set of embedded steering strategies was also required for the car to automatically move along the planned paths and directions. In order to calculate the car's relative distance and angle against the target point, an electronic compass H was used. Also, the heading angle of the servo motor was set as θ_s. When $\theta_s = 0$, the robot car would go straight. When the θ_s value was positive, the robot car would turn right. When the θ_s became negative, the robot car would turn left. The maximum range of the angle was set at positive/negative 180°.

The input latitude and longitude data of the robot car (lat_r, lng_r) and the target point (lat_g, lng_g), which were received from the GPS, were used in Equation (12) to get $\varnothing$ (the angle of the target point against the exact north). Then we deducted H, the heading direction (see Equation (13)), to get the direction error θ_t, which was the angle between the current direction and the target point. We then used the proportional control θ_t to multiply K_P (the gain used for servo proportional control; $K_P = 0.161$) after adding θ_c (the center angle of the servo motor for the steering) to obtain θ_s (that is, the command sent to the servo motor for moving). In addition, the maximum range for the motor to move, ±15°, was set to prevent the cars from rolling over under high-speed turning. The $K_P\theta_t$ was determined with the understanding it should not exceed ±15° (Equation (14)).

$$\varnothing = \tan^{-1}\left(\frac{lng_g - lng_r}{lat_g - lat_r}\right) \tag{12}$$

$$\theta_t = \varnothing - \mathrm{H} \tag{13}$$

$$\begin{cases} \theta_s = \theta_c + K_P\theta_t \\ K_P\theta_t = 15, \quad K_P\theta_t \geq 15 \\ K_P\theta_t = -15, \quad K_P\theta_t \leq -15 \end{cases} \tag{14}$$

In addition to steering, determining whether to reach the target point was equally important. Once the car reached the target point, only then was it allowed to move to the next one. The distance between the car and the target point was the key. We set the target point radius R. If the robot car was entering the range of R (the distance between the robot car and the target point < R), the target point hit would be determined. In this case, R was set with an appropriate value, understanding that it would be difficult for the car to hit the target points with too small or too large of an R value. The R value, then, was acknowledged as affecting the accuracy of the experiment [22].

5. Tests and Results

This section describes the tests for the proposed hybrid path programming method with maximum range constraint for the mission planning of multi-robot cars. Three types of tests were performed:

1. Convergence Test—This was conducted to confirm the convergence of the hybrid programming algorithm under range limitation.
2. Bench Test—This was performed to verify competitiveness by comparing with existing public TSPLIB instances.
3. Field Test—This was used to validate the practicality of the proposed algorithm by a group of three robot cars deployed on a field test.

In all tests, the proposed hybrid path programming computer code was processed on the remote control station based on a laptop PC with Intel Core i7 2.4 GHz CPU and 8 GB RAM.

5.1. Convergence Test

To see whether this hybrid path programming algorithm could successfully converge a solution set to optimize the route of each robot car group under their maximum travelable distance limitation, a series of tests were conducted. First, we ensured the algorithm convergence status by a simple 22-point test. As shown in Figure 9, a solution was converged at the 63rd generation. However, the solution was not immediately improved at the first generation. The solution exceeded the maximum distance limit as well as the Candidate List $= N(x_i) - T - S_{lmt} = \varnothing$, so we followed the improved tabu search and liberalized the Candidate List = $N(x_i) - T$. We compared the maximum distance of each of the paths and selected the solution with the minimum value in the Candidate List. Due to the maximum distance limit, there was a function to restrict the maximum distance of the feasible solution, which was not allowed to exceed this threshold. Starting from the maximum distance, we picked the minimum maximum distance of the neighborhood solution, and then made the current solution gradually close to the threshold (maximum distance limit). This eventually sufficed for meeting the threshold. Finally, the current solution was eligible (see the green line in Figure 9—the distance variation of the current solution). In this way, the current solution was improved and gradually got close to the feasible solution.

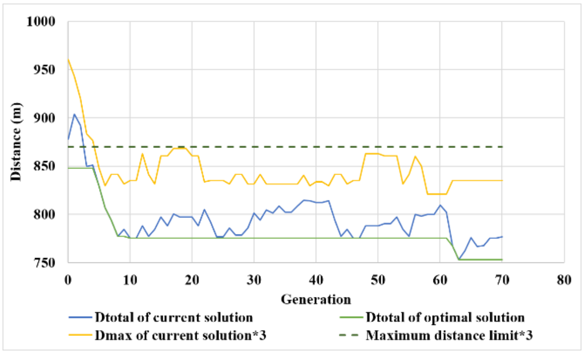

Figure 9. Solution Convergence Diagram.

5.2. Bench Test

The problem definition of this research is similar to mTSP. The difference, in comparison to mTSP, is that this research is limited to the travel distance of each robot car. Since mTSP instances are easy to obtain, we used the innovative algorithm developed in this research to solve the same mTSP problem for comparing.

Most scholars have modified TSPLIB instances to test mTSP, because mTSP does not have public instances. In this research, Pr76, Pr152, Pr226, Pr299 and Pr439 were tested. The mTSP rule that each salesman must visit more than two targets was used. We then compared with MGA [23], MACO [24], NMACO [11] and SA+EAS [25]. For establishing the initial solution, the tabu list length was set to 30, and the condition set for stopping was that the optimal solution was not updated, and it improved in 50 generations. The tabu list length of Improvement of Each Single Path was set to 50. The tabu list length of Improvement between Different Paths was 50 as well. As well as this, the condition set for stopping calculations for programming both paths was that the solution had neither been updated nor improved in the most recent 10 generations. The condition set for stopping the path improvement part was that the optimal solution had not been updated or improved in the second iteration.

Table 1 is the comparison result of our algorithm in TSPLIB instances with each other. The 2TS+2OPT was the hybrid algorithm developed in this research. The number of the target is expressed as n. The number of salesmen is shown as m. The maximum number of waypoints (cities) that each vehicle (salesman) could visit is denoted by u. As a result, the distance values of 2TS+2OPT are better than the others, and most of the CPU Times are less than the others as well.

Table 1. The Comparison Result of 2TS+2OPT Algorithm in the library of sample traveling salesman problem (TSPLIB) Instances.

Name	pr76	pr152	pr226	pr299	pr439
n	76	152	226	299	439
m	5	5	5	5	5
u	20	40	50	70	100
SA+EAS	157482	127755	167655	81922	161698
NMACO	157413	127781	167239	81261	160298
MACO CPU Time(s)	178597 51	130953 128	167646 143	82106 288	161955 563
MGA CPU Time(s)	178597 43	130953 91	167646 165	82106 363	173839 623
2TS+2OPT **CPU Time(s)**	**153840** **11.3**	**121165** **51.2**	**159831** **153.4**	**72813** **190.5**	**141526** **455.4**

5.3. Field Test with Multiple Robot Cars

In this study we set out to solve the path programming of a multi-target wide area. Subject to vehicle ability constraints, cars were not able to travel to all target points. Therefore, we sent multiple vehicles to respectively travel to target points and complete tasks. We selected a site in Yunlin, Taiwan, and set up several target points on it. With a maximum distance limit set up, the shortest total distance paths and the total distance limits for each robot car were set, so they did not exceed this limit. A hybrid tabu search combined with a 2-opt swap method was adopted to program the optimal path in the Remote Monitoring Station, which was a laptop with CI7, 2.4 GHz, and 8 GB RAM. For establishing the initial solution, the tabu list length was set to 30. The condition set for stopping was that the optimal solution had not been updated nor improved in 50 generations. The tabu list length of Improvement of Each Single Path was set to 30. The tabu list length of Improvement between Different Path parts was 30. Additionally, the condition set for stopping calculations for the programming of both paths was that the solution had not been updated or improved in the most recent 50 generations. The condition set for stopping the path improvement part was that the optimal solution had not been updated nor improved in the second iteration. After all of these settings were in place, the paths were assigned to robot cars for them to run on the designated site with the paths programmed.

5.3.1. Maximum Distance Limit = 170 m

The test started by randomly setting up 15 target points on the e-map of the Remote Monitoring Station. We assumed that the maximum travelable distance limit was 170 m. If we were sending a single robot car to travel all target points, the optimal (shortest) path was 270.6 m (as shown in Figure 10). But this path obviously had exceeded the maximum distance limit that a single car can handle for completing a task; hence three cars were sent for this test. The system soon provided new paths (as shown in Figure 11), with the shortest distance by using the algorithm developed in this research. In this test, the CPU time of the Remote Monitoring Station computer was only 0.79 s. With this solution, the A-path was 81.3 m; the B-path was 164.7 m; and the C-path was 139.9 m. None of the paths exceeded the maximum distance limit. The shortest total distance was 385.9 m. Then the system automatically dispatched those three routes to robot cars to visit their assigned target points. We recorded the path trajectory of each robot car, as shown in Figure 12. The actual total distance was 401.1 m. The actual distance of the A path was 86.9 m; for the B path it was 167.9 m; and for the C path it was 146.3 m. The maximum error of all robot cars was 6.8% of car A. The error of total distance was 3.9% (Table 2). This error was mainly caused by GPS drifting, a bumpy surface and steering center offset factors.

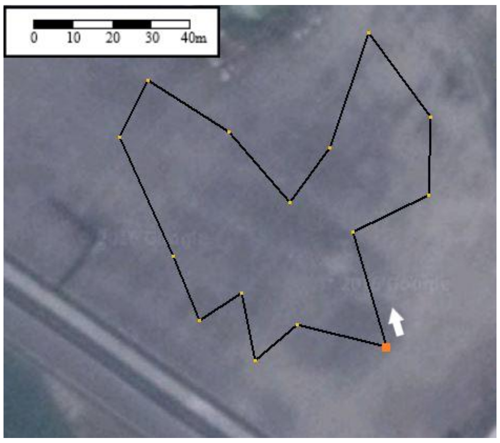

Figure 10. Shortest Single Distance Path.

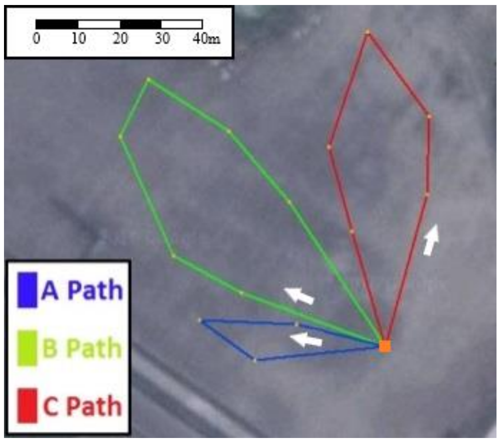

Figure 11. Shortest Total Distance Path When the Maximum Distance Limit Is 170 m.

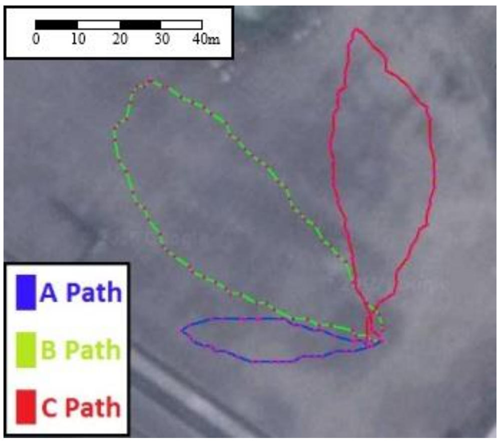

Figure 12. Experimental Trajectory When the Maximum Distance Limit Is 170 m.

Table 2. The Comparison of Shortest Total Distance (Theoretical) and Total Experimental Trajectory (Actual) under the Constraint of the Maximum Distance Limit of 170 m for Each Car.

	A	B	C	Total
Theoretical	81.3	164.7	139.9	385.9
Actual	86.9	167.9	146.3	401.1
Error	+5.6	+3.2	+6.4	+15.2
Error%	+6.8%	+1.9%	+4.5%	+3.9%
Path programming CPU time = 0.79 s				

5.3.2. Maximum Distance Limit = 164 m

With the same location target points set up, we tuned down the maximum distance limit as 164 m. The algorithm converged a new solution set (as shown in Figure 13) in a very short (0.75 s) CPU time. For this solution, the A path was 105.2 m; the B path was 159.9 m; and the C path was 139.9 m. None of paths exceeded the maximum distance limit. The total distance, however, increased to 405 m, because the maximum travelable distance of each car was compressed, which made the solution more "load-balanced". The completion time was relatively less. After the solution was converted, the system immediately dispatched those three routes to robot cars to visit their assigned target points. We recorded the path trajectory of each robot car, as shown in Figure 14. The actual total distance was 412.5 m. The actual distance of the A path was 106.8 m; for the B path it was 160.9 m; and for the C path it was 144.8 m. The maximum error of all robot cars was 3.5% of car C. The error of total distance was 1.8% (see Table 3). It is worth mentioning that none of the robots exceeded the range limitation.

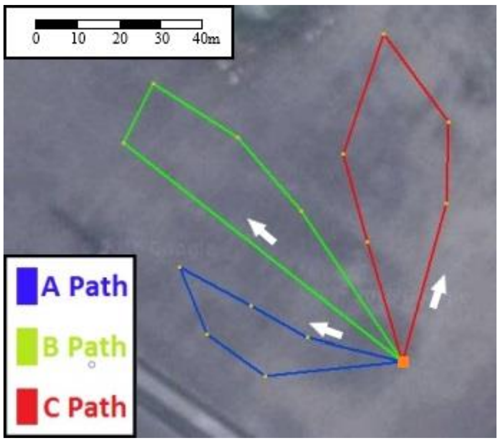

Figure 13. Shortest Total Distance Path When the Maximum Distance Limit Is 164 m.

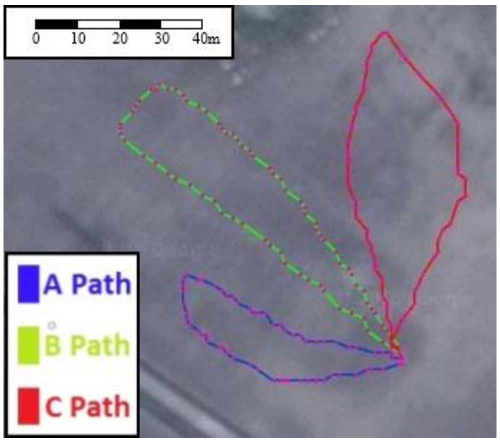

Figure 14. Experimental Trajectory When the Maximum Distance Limit Is 164 m.

Table 3. The Comparison of Shortest Total Distance (Theoretical) and Total Experimental Trajectory (Actual) under the Constraint of Maximum Distance Limit 164 m for Each Car.

	A	**B**	**C**	**Total**
Theoretical	105.2	159.9	139.9	405.0
Actual	106.8	160.9	144.8	412.5
Error	+1.6	+1.0	+4.9	+7.5
Error %	+1.5%	+0.6%	+3.5%	1.8%
Path programming CPU time = 0.75 s				

5.4. Discussion

From the bench test, compared to other algorithms, the 2TS+2OPT hybrid algorithm proposed in this research has very high advantages in both path optimization and CPU time, which are crucial for the practical applications of unmanned system. Besides, when examining the results generated from this hybrid algorithm of path programming, it was found that the total distance was inversely proportional to the maximum distance limit value. Thus, the lower the maximum distance limit value was, the longer the total distance would be. Relatively speaking, the lower the maximum distance limit value, the shorter the mission time. In general, a tighter margin in onboard energy capacity yielded a "load-balanced" situation for each robot in the group, which means every robot had equal loading.

From the field experiments, we obtained the error of maximum total distance at 3.9%, and the maximum error of each robot car at 6.8%. These are both minor and representative of a rather satisfying result, as we expected. In addition, we also found a few minor errors in Algorithm 1; Table 1, which were due to the following three factors:

1. GPS drifting

The GPS device adopted in this research was designed for general commercial purposes, with couple meters measuring error. This GPS drifting may be easily fixed with higher-level equipment, such as that of a real-time, kinematic GPS.

2. Pavement condition

The paths programmed by the system were generated based on a smooth ground surface for car operation. But in fact, there some unexpected surface conditions, like tiny pebbles on the pavement that may have caused an offset to the route.

3. Offset of steering center

The steering mounted on the robot car consisted of a servo motor and a steering mechanism; this unit may have been offset while it was traveling among target points during a long run. This offset might have somehow led to the car moving off the path, and thus resulted in a few minor errors at the end. In this research, we tried to minimize the effect of this factor by regularly correcting the steering.

In terms of the case that set the maximum distance limit at 170 m, since the longest path was 164.7 m (which was very close to the maximum distance limit), the robot car could run and exceed the limit as long as any error occurred. For example, the car with a maximum error of 6.8% in this experiment was obviously not able to complete the task. Thus, reserving 10% as an allowance margin when setting the maximum distance limit is recommended (maximum distance limit: 10%).

6. Conclusions

In this research, a hybrid 2TS+2OPT algorithm with limited range constraints for the path programming of multiple robot vehicles was successfully developed. This innovative algorithm is superior in both better path optimization and shorter CPU time, compared to other mTSP algorithm solutions. In the last stage of this research, a general scenario was presented to show the whole process of the multi-robot mission planning, in which three robots were deployed in a field to complete a series of wide area multi-waypoint tasks which were far beyond a single car's endurance capabilities. Those tests validated that the algorithm can successfully optimize robots' routes to visit assigned target points within their range limitations.

In real-world, unmanned vehicles practices, optimized paths based on onboard energy capacity (fuel or battery) constraints are critical to multiple-agent system applications of this type, including large area multi-points surveillance exercises, robot swarm deliveries, multi-drone attacks, and so on. This research could contribute to those types of instances.

Author Contributions: Conceptualization, M.-T.L.; Formal analysis, B.-Y.C., M.-T.L. and Y.-C.L.; Methodology, M.-T.L.; Project administration, M.-T.L.; Software, B.-Y.C. and M.-T.L.; Writing—original draft, B.-Y.C. and M.-T.L.; Writing—review & editing, M.-T.L. and Y.-C.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Technology, Taiwan (ROC), under grant number MOST-107-EPA-F-010-001.

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

References

1. Luan, Y.; Xue, H.; Song, B. The Simulation of the Human-Machine Partnership in UCAV Operation. In Proceedings of the 26th International Congress of the Aeronautical Sciences, Anchorage, AK, USA, 14–19 September 2008.
2. Cai, B. Path Programming for Autonomous Unmanned Vehicle System. Bachelor's Thesis, National Formosa University (NFU), Huwei, Taiwan, 2015.
3. Atanasov, N.; Ny, J.L.; Daniilidis, K.; Pappas, G.J. Decentralized Active Information Acquisition: Theory and Application to Multi-robot SLAM. In Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, 26–30 May 2015; pp. 4775–4782.
4. Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Emaru, T. Hitchhiking Robots: A Collaborative Approach for Efficient Multi-Robot Navigation in Indoor Environments. *Sensors* **2017**, *17*, 1878. [CrossRef] [PubMed]
5. Maity, D.S.; Goswami, S. Multipath Data Transmission with Minimization of Congestion Using Ant Colony Optimization for mTSP and Total Queue Length. *IJLRST* **2013**, *2*, 109–114.
6. Sariel-Talay, S.; Balch, T.R.; Erdogan, N. Multiple Traveling Robot Problem: A Solution Based on Dynamic Task Selection and Robust Execution. *IEEE/ASME Trans. Mechatron.* **2009**, *14*, 198–206. [CrossRef]
7. Ahmadvand, M.; Yousefikhoshbakht, M.; Mahmoodi Darani, N. Solving the Traveling Salesman Problem by an Efficient Hybrid Metaheuristic Algorithm. *JACR* **2012**, *3*, 75–84.
8. Atashpaz-Gargari, E.; Lucas, C. Imperialist Competitive Algorithm: An Algorithm for Optimization Inspired by Imperialistic Competition. In Proceedings of the IEEE CEC 2007, Singapore, 25–28 September 2007.
9. Larki, H.; Yousefikhoshbakht, M. Solving the Multiple Traveling Salesman Problem by a Novel Metaheuristic Algorithm. *JOIE* **2014**, *16*, 55–63.
10. Dorigo, M.; Maniezzo, V.; Colorni, A. Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **1996**, *26*, 29–41. [CrossRef] [PubMed]

11. Yousefikhoshbakht, M.; Didehvar, F.; Rahmati, F. Modification of the Ant Colony Optimization for Solving the Multiple Traveling Salesman Problem. *ROMJIST* **2013**, *16*, 65–80.
12. Necula, R.; Breaban, M.; Raschip, M. Performance Evaluation of Ant Colony System for the Single-Depot Multiple Traveling Salesman Problem. In Proceedings of the 10th International Conference on Hybrid Artificial Intelligence Systems, Bilbao, Spain, 22–24 June 2015; pp. 257–268.
13. Xu, M.; Li, S.; Guo, J. Optimization of Multiple Traveling Salesman Problem Based on Simulated Annealing Genetic Algorithm. *MATEC Web Conf.* 2017. [CrossRef]
14. Côté, J.F.; Potvin, J.Y. A Tabu Search Heuristic for the Vehicle Routing Problem with Private Fleet and Common Carrier. *EJOR* **2009**, *198*, 464–469. [CrossRef]
15. Liao, T.; Huang, W. A Study of Dynamic Logistics Based on Two-Phased Method. *IJAIT* **2008**, *2*, 76–94.
16. Lin, W.; Cheng, F.; Tsay, M. An Improved Tabu Search for Economic Dispatch with Multiple Minima. *IEEE Trans. Power Syst.* **2002**, *17*, 108–112. [CrossRef]
17. Bektas, T. The Multiple Traveling Salesman Problem: An Overview of Formulations and Solution Procedures. *Omega* **2006**, *34*, 209–219. [CrossRef]
18. Hung, F. Vehicle Routing Problem of Integrated Supply Medical Materials in Strategic Alliance Hospitals—A Study of One Medical Center. Master's Thesis, National Yunlin University of Science and Technology, Yunlin, Taiwan, 2004; p. 15, Unpublished.
19. Lin, S. Computer Solutions of the Traveling Salesman Problem. *BSTJ* **1965**, *44*, 2245–2269. [CrossRef]
20. Khamis, A.; Hussein, A.; Elmogy, A. *Multi-robot Task Allocation: A Review of the State-of-the-Art, Cooperative Robots and Sensor Networks 2015*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 31–51.
21. Balch, T.R. Social Entropy: A New Metric for Learning Multi-Robot Teams. In Proceedings of the 10th International FLAIRS Conference (FLAIRS-97), Daytona, FL, USA, 11–14 May 1997.
22. Lai, L.; Hsieh, Y. The Research in Real-time Dynamic Path Programming for Multiple Autonomous Ground Vehicle Mission. Bachelor's Thesis, National Formosa University (NFU), Huwei, Taiwan, 2016.
23. Tang, T.; Liu, J. Multiple Traveling Salesman Problem Model for Hot Rolling Scheduling in Shanghai Baoshan Iron & Steel Complex. *EJOR* **2000**, *24*, 267–282.
24. Junjie, P.; Dingwei, W. An Ant Colony Optimization Algorithm for Multiple Traveling Salesman Problem. In Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC '06), Beijing, China, 30 August–1 September 2006; pp. 210–213.
25. Yousefikhoshbakht, M.; Sedighpour, M. A Combination of Sweep Algorithm and Elite Ant Colony Optimization for Solving the Multiple Traveling Salesman Problem. *P. Pomanian. Acad. A* **2012**, *13*, 295–302.