2008 American Control Conference
Westin Seattle Hotel, Seattle, Washington, USA
June 11-13, 2008

Staying-Alive and Energy-Efficient Path Planning for Mobile Robots

Tianmiao Wang[1], Bin Wang[1], Hongxing Wei[1], Yunan Cao[1], Meng Wang[2], Zili Shao[2*]

[1]The Robot Research Institute
Beihang University
Beijing 100083, China
{itm, wbin, whx}@me.buaa.edu.cn

[2]Department of Computing
The Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
cszlshao@comp.polyu.edu.hk

Abstract—**As most mobile robots are powered by batteries, their energy and operation times are limited. Therefore, how to minimize energy consumption and keep mobile robots to stay alive becomes an important problem. In this paper, by applying a dynamic energy-evaluation scheme, in which we consider if a robot has enough energy to go to next location, finish the task and return to the docking station in the path planning, we propose two staying-alive and energy-efficient path planning approaches based on the greedy TSP and Tabu-search methods, respectively. The experimental results show that our Tabu-search-based approach is the best and can provide an effective path planning by which a robot can be guaranteed to stay alive and finish all tasks with the minimum energy.**

I. Introduction

MOBILE robots can be used in many missions such as elder care, shopping navigation, carpet cleaning, lawn moving and rescue assistance after disasters [1][2]. As most mobile robots are powered by batteries, their energy and operation times are limited. For example, the humanoid robot from Honda can only walk for approximately 30 minutes with its rechargeable battery [14]. Therefore, how to minimize energy consumption and keep mobile robots to stay alive becomes an important problem. To stay alive, we need to guarantee that if a robot does not have enough energy, he should be able to return to a docking station for battery changing or recharging, which needs to be done in an energy-efficient way for energy saving. In this paper, we focus on solving the staying-alive and energy-efficient path planning problem: given a robot and various tasks in different locations, how to find a path planning with the minimum energy to finish all tasks and stay alive.

Energy-efficient path planning has been extensively studied from the previous work. In [9], Katoh et al propose an energy-efficient motion planning method for space manipulator by controlling the motion of the space manipulator to be elliptic. In [10] [11], Mei et al analyze power consumption of a robot at different speeds and propose an effective energy-aware motion scheme. Barili et al [12] develop an energy-saving scheme by controlling the speeds and avoiding unnecessary stops for mobile robots. Chong et al [8] propose minimum-energy velocity-trajectory-control scheme considering practical energy consumption dissipated in motors. In [15], Jia et al propose a cost-efficient motion planning algorithm by integrating grid and topological information for robot exploration. In all of the above work, however, staying alive is not considered. As we show later, in a path planning for mobile robots powered by batteries, if returning to a docking station for battery changing or recharging is not considered, a robot may exhaust all of its energy and stop in the middle of the path.

To solve the staying-alive problem for mobile robots, several methods have been proposed [5-7]. Seungjun Oh et al [7] implement the auto recharging device on a mobile robot. However, in these methods, path planning is either not considered or handled with a static manner. As we show in Section 2, a static energy lower bound method may not work very well for energy saving. In [18-19], Zebrowski et al propose an energy delivery approach called a tanker approach. In their approach, a tanker robot serving as "mother" robot to traverse and distribute energy cells to "worker" robots if demanded. In [20], Floreano et al propose an evolution of a discrete time recurrent neural network approach that allows the robot to choose trajectory as function of location and remaining energy. Our work is a good complement for the above methods by providing staying-alive and energy-efficient path planning so robots or tanker robots can utilize the routes generated to finish their tasks with the minimum energy.

In this paper, we propose two approaches to solve the staying-alive and energy-efficient path planning problem. To guarantee that a robot can always return to the docking station for battery changing or recharging, we apply a dynamic energy-evaluation scheme, in which we consider if a robot has enough energy to go to next location, finish the task and return to the docking station in the path planning. Based on this scheme, we propose two approaches for energy minimization. As the problem is a variation of the traveling salesman problem (TSP), a well-known NP-complete problem, we first propose an approach based on a greedy TSP method [22]. Then we propose a better approach based on the Tabu-search

* The corresponding author.

method [17]. We conduct experiments based on a simulation environment. The experimental results show that our Tabu-search-based approach can provide an effective path planning by which a robot can be guaranteed to stay alive and finish all tasks with the minimum energy.

The rest of the paper is organized as follows. Motivational examples are given in Section II. In Section III, we introduce the basic concepts. In Section IV, we provide our approaches. The experimental results are provided in Section V, and the conclusion is shown in Section VI.

II. Motivating Examples

In this section, we motivate the staying-alive and energy-efficient path planning problem by showing the different path planning with different energy consumption from different method.

Given an environment shown in figure 1 in which there are eleven tasks in (A to K) locations, each tasks with a value which denotes energy require for a robot doing the task. The robot needs to traverse all tasks starting from and back to the docking station. The distance between tasks can be obtained by Dijistra algorithm [21]. We assume that robot consumes 1 unit of energy on 1 unit distance, and robot has 1 thousand units of energy after recharged. These assumptions are only for demonstration purpose. Our technique is general enough to deal general energy models as discussed in later sections. Figures 1, 2, 3 show the different routes with different path planning methods.

In Figure 1, the route is obtained based on the TSP greedy approach [22], in which the robot traverses all tasks by picking and going to a task that is the closest to him. When robot finished task at I, energy left is 26.20 units, and robot energy exhausted after arrived at J.

This problem can not be solved very well with a static manner. For example, using the method with a static energy-evaluation scheme [7], suppose that a robot is required to return to the docking station if his energy level is reach to a threshold such as 10% of robot battery capacity. As figure 2 shows the robot reaches to H' and find that his energy level reaches to the threshold. So he returns to the docking station, and then continues to finish all tasks after battery changing or recharging. As we can see, H' is far from the docking station, the routes from H' to the docking station and from the docking station to I is very long, so more energy is consumed.

Figure 3 shows the route obtained by our method based on the dynamic energy-evaluation scheme. In our method, we combine the path planning and staying alive together. When we do path planning, we will consider if the robot has enough energy to go to next location, finish the task and return to the docking station. Therefore, at H, the robot will return to the docking station and then continue. In this way, the robot can finish all tasks with the minimum energy and staying alive.

From these examples, we can see that neither energy-efficient path planning methods without considering staying alive nor static energy lower bound methods can solve

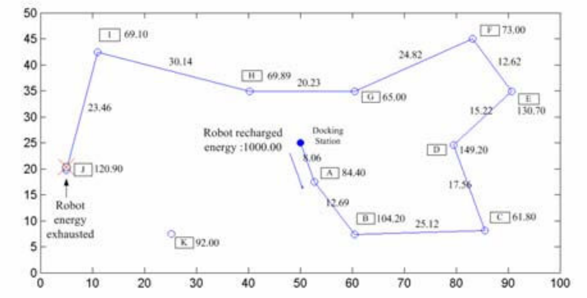

Figure.1 Path planning without the consideration of staying alive, robot exhausts when doing task in location J.

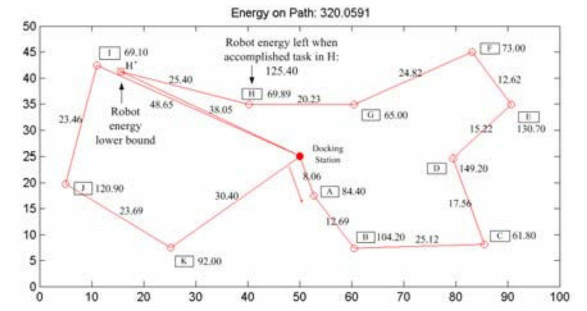

Figure.2 Path planning using static energy-evaluation scheme.

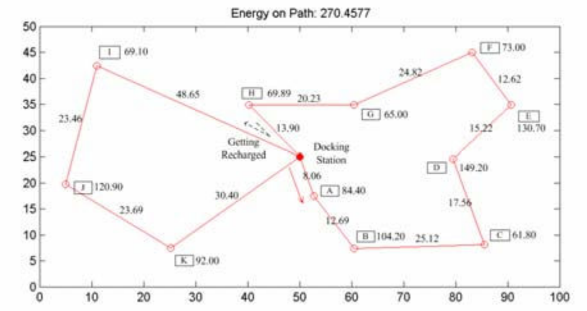

Figure.3 Path planning using dynamic energy-evaluation scheme.

this problem very well. Therefore, we propose a dynamic energy-evaluation method to solve this problem.

III. Basic Concepts and Problem

In this section, we introduce the basic concepts and formally define the problem in Section III-A and Section III-B, respectively.

A. Graph Model

We use a node-weighted edge-weighted directed graph to model the staying-alive and energy-efficient path planning problem.

Let $G=(V, A, W, E)$ be a directed graph where $V=\{v_0, v_1, \ldots, v_n\}$ is the vertex set, $A=\{a(v_i \rightarrow v_j): i \neq j\}$ is the edge set, $W=\{c_{i,j} : v_i \rightarrow v_j \in A\}$ is the weight of each edge, $E=\{Ev_1, \ldots, Ev_n\}$ is the energy cost for finishing the task. In the node set V, v_0 denotes the docking station and $v_i \in \{v_1, \ldots, v_n\}$ represents the location where Task i occurs in the environment. In the set of

W, $c_{i,j}$ is a positive number to represent the energy cost when the robot moves from v_i to v_j.

B. Problem Definition

The staying-alive and energy-efficient path planning problem for mobile robots is formally defined as follows:

1) The robot needs to traverse all task nodes $\{v_{1\ldots}\ v_n\}$ starting from and back to v_0 (the docking station);

2) Each node in $\{v_{1\ldots}\ v_n\}$ needs to be visited once, and the corresponding task is finished after the node has been visited;

3) When the robot returns back to the docking station, the remaining energy must be greater than $E_{docking}$ which is the energy required to finish operations for battery changing or recharging [5-6].

C. The Tabu-Search Method

The defined problem is a variation of the traveling salesman problem, a well-known NP-complete problem [22]. In particular, our problem is close to the VRP (Vehicle Routing Problem) problem what can be solved by the Tabu-search method [14, 17, 18]. The Tabu-search method is a generic method, in this paper, we define our tabu candidate selection strategies, the objective function with penalty factor for staying-alive and tabu properties to solve our problem.

IV. STAYING-ALIVE AND ENERGY-EFFICIENT PATH PLANNING

In this section, we propose our dynamic-energy-evaluation –based method to solve the staying-alive and energy-efficient path planning problem. We first introduce our scheme in Section IV-A. Then we propose two algorithms for energy optimization based on this scheme in Sections IV-B and IV-C, respectively.

A. Dynamic Energy-Evaluation Scheme

As shown in Section II, the scheme of fixing an energy lower bound for robot to return cannot achieve big energy saving. Thus, we propose a dynamic energy-evaluation scheme to solve this problem. Our basic idea is to dynamically evaluate whether a robot needs to return back to the docking station before it finishes the current task or not. The lower bound energy is calculated as follows,

$$E_{lower\ bound} = E_{(v_{i-1},v_i)} + E_{v_i} + E_{(v_i,v_0)} + E_{docking} \qquad (1)$$

in which, $E_{(vi\text{-}1,vi)}$ is the energy requirement for going to the task position v_i from current position $v_{i\text{-}1}$ (for i≥1); Ev_i is the energy cost for finishing the task at location v_i.; $E_{(vi\text{-}1,v0)}$ is the energy requirement for returning to the docking station v_0 from task position v_i; $E_{docking}$ is the energy required for battery changing or recharging in the docking station.

If the current energy of the robot is greater than the lower bound, the robot can finish the next job and return to the docking station. Otherwise, the robot should go back to the docking station for battery changing and recharging.

B. The TSP-Greedy-Based Algorithm (GTSP)

As our problem is a variation of the TSP problem, thus, we propose an algorithm that combines the TSP greedy algorithm [22] and our dynamic energy-evaluation scheme. The GTSP algorithm is shown in Algorithm 1.

```
Algorithm 1. Algorithm GTSP_Generation
Input: A graph G= (V, A, W, E) (defined in Section III-A).
Output: Staying alive path P_GTSP.
1: Generate a greedy based path without energy consideration according to [22].
2: Calculate the energy need of adjacent vertexes.
3: Route from docking station v_0 with the dynamic energy-evaluation scheme.
```

In GTSP, we first obtain a path planning based on the TSP greedy algorithm. And then we evaluate the path by calculating the lower bound energy, $E_{lower\ bound}$. Starting from v_0 (the docking station), for each node in the path, if the energy that the robot has is greater than $E_{lower\ bound}$, then the robot goes ahead to the next node; otherwise, the robot goes back to the docking station and continues the unfinished tasks after battery changing or recharging.

C. The Tabu-Search-Based Algorithm (TS)

In this section, we first propose an algorithm that is based on the TS method. Then we introduce the details of its two key functions in Section IV-C-1 and Section IV-C-2, respectively. The TS algorithm is shown in Algorithm 2.

The following notation is used in the presentation. Based on our initial solution, we get r sub routes. The energy consumption for routing the path is written as follows:

$$E_1(P) = \sum_r \sum_{(v_i,v_j)\in R_r} c_{ij} \qquad (2)$$

The objective function, $E_2(P)$, is defined as follows,

$$E_2(P) = E_1(P) + \alpha \cdot \sum_r \left[E_{(v_i,v_j)\in R_r} + E_{(v_i)\in R_r} + E_{docking} - E_{capicity} \right]^+ \qquad (3)$$

in which, $E_{(v_i,v_j)\in R_r}$ is the energy requirement for the route r; $E_{(v_i)\in R_r}$ is the energy requirement for finishing the tasks in each sub route r; $[x]^+$=max(0,x) and α is a penalty coefficient.

From Equ (3), we can see that if the robot has enough energy to traverse the path and return to the docking station, then $\left[E_{(v_i,v_j)\in R_r} + E_{(v_i)\in R_r} + E_{docking} - E_{capicity} \right]^+$ equals zero. Otherwise, it implies that the robot does not have enough energy to fulfill the path; thus, a positive value is added to $E_1(P)$ to denote the penalty.

Based on this objective function, our Tabu-Search-Based algorithm is shown in Algorithm 2.

```
Algorithm 2. Algorithm TS-STSP
Input:   A graph G= (V, A, W, E). (defined in Section III-A)
Output:  Tabu-search based staying alive path P_TS,
Energy consumption on path P_TS: E_1(P_TS).
/*Step 1: Generate the sequential-based initial solution */
1: Call function STSP_Generation (G) to generate the staying alive path P_STSP in sequential order of angle.
/*Step 2: Tabu-search optimization */
```

```
2: Call function TS_Optimization ($G$, $P_{INITIAL}$) to optimize initial solution.
```

The Tabu-Search-Based algorithm consists of 2 steps. The first phase is to call function **STSP_Generation ()** to obtain a good initial solution. The second phase is to call function **TS_Optimization ()** to perform tabu-search optimization. **STSP_Generation ()** and **TS_Optimization ()** are shown in Algorithm 3 and Algorithm 4, respectively, which are presented below.

1) Initial Solution Generation

Initial solution selection can influence the performance and convergence rate of tabu search [18]. We have two initial solutions for tabu search; one is for improved greedy based method (GTSP), the other is a sequential method (STSP) which we will introduce in this section.

The sequential staying-alive method is shown in Algorithm 3 with the following steps:

Step 1 Let the docking station be the origin. All tasks are labeled according to the angle. For example, as shown in Figure 4, v_1 is first task from the horizon line with angle θ.

Step 2 Calculate the energy consumption between two adjacent vertexes with the sequence obtained in Step 1.

Algorithm 3. Algorithm **STSP_Generation**

```
Input:   A graph $G = (V, A, W, E)$.
Output: Staying alive path $P_{STSP}$.
1:  Label vertex set {$v_{1,\ldots,}$ $v_n$} according to the angle.
2:  Calculate the energy need of adjacent vertexes in Step 1.
3:  Find two vertexes ($v_i$, $v_{i+1}$) that form the maximum angle.
4:  Route from the $v_i$ and $v_{i+1}$ respectively, with the dynamic energy-evaluation scheme.
5:  Compare the energy consumption of two routes and pick up the less energy consumption one as $P_{STSP}$.
```

Step 3 Find two vertexes (v_i, v_{i+1}) which form the maximum angle in the plane. As shown in figure 5, nodes (v_1, v_2) have the maximum angle.

Step 4 The route follows the vertex sequence clockwise (v_i, $v_{i-1,\ldots,}$ v_1, v_n, v_{n-1},…, v_{i+1}) and anticlockwise (v_{i+1}, $v_{i+2,\ldots,}v_n$, v_1, v_2,…, v_i), respectively, with the dynamic energy-evaluation scheme. As shown in figure 5, route the vertex sequence clockwise {v_1, $v_{12,\ldots,}$ v_3, v_2} and anticlockwise {v_2, $v_{3,\ldots,}$ v_{11}, v_{12}, v_1}, respectively.

Step 5 Compare the energy for two routes, and pick up the route with less energy.

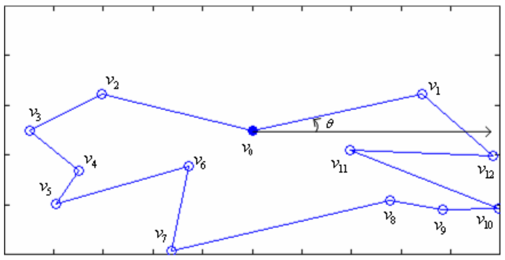

Figure.4 Labels of task locations according to the angle in anticlockwise

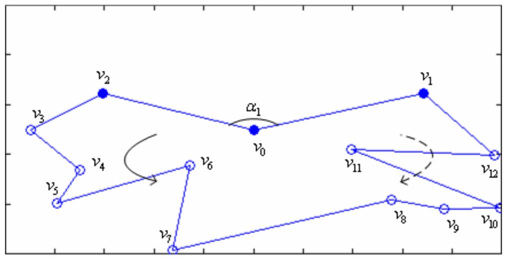

Figure.5 Adjacent nodes (v1, v2) have the maximum angle.

2) Tabu Search Optimization

In this section, we propose a Tabu-search optimization method based on the initial solutions we obtained from the above steps. The algorithm is shown in Algorithm 4.

Algorithm 4. Algorithm **TS_Optimization**

```
Input:   A graph $G = (V, A, W, E)$ and initial staying alive path $P_{INITIAL}$.
Output: Tabu-search based staying alive path $P_{TS}$, energy consumption on the path $E_1(P_{TS})$.
0:  Take $P_{INITIAL}$ as the input solution, set iter to 1.
1:  while iter is less than MAX_ITERATION_NUM
2:     Generate the candidates according to input solution.
3:     Calculate the objective function in Equ (3) for candidates.
4:     /* Aspiration criterion */
5:     if the best candidate better than the best solution so far.
6:        Select the candidate as next input solution.
7:     else
8:        Select the best candidate which is not tabooed in the
            tabu list as next input solution.
9:     end if
10:    Update tabu list
11:    /* Stop criterion: break from circulation */
12:    if iter is larger than $\sqrt{\text{MAX\_ITERATION\_NUM}}$ and the change of ten latest best so far solution is less than 0.1%
13:       break
14:    end if
15: end while
```

Our Tabu-search optimization algorithm consists of following steps:

Step 1. (Candidates generation) Generate the candidates according to input solution. Candidates consist of the solutions obtained by performing all possible moves according to input solution. The swapping and insertion are used as moving strategy, and swapping is shown in figure 6. The swapping strategy is implemented on the tasks. We prohibit the swapping between a task and the docking station that is part of insertion strategy. The insertion strategy is implemented on the locations of tasks between inner and inter sub routes as well.

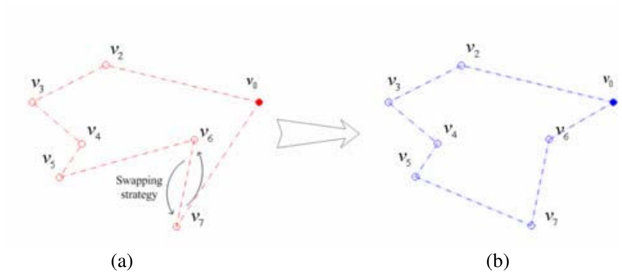

Figure.6 Swapping strategy is used to generate the neighbor space for tabu search.

Step 2. (Objective function calculation) Calculate the objective function in Equ (3) for candidates generated in the former step. Sort candidates with the ascending order based on the values are obtained from the objective function.

Step 3. (Aspiration criterion) If the best candidate can achieve a better solution, then we update the best solution no matter whether the candidate is tabooed. And then we select the solution as the next input. Otherwise, we select the best candidate which is not tabooed in the tabu list as the next input solution.

Step 4. (Update tabu list) Record the executed move as tabu moves in the tabu list with tabu length, and we reduce 1 tabu length of tasks tabooed before. When it becomes zero, the tabooed move can take place freely.

Step 5 (Stop criterion) Check the stop criterion. If the variation of the objective function for scheme "best so far" is less than a given value in several times, then we stop the search. Otherwise, the algorithm returns to Step 2 and continues the tabu search procedure. As shown in figure 7, we achieve significant improvement on the optimal solution according to the initial solution. The scheme "best solution so far" holds the optimal solution in the input solutions, and terminates the tabu-search when it satisfies the stop criterion.

Figure 8 is the best solution according to our tabu-search method. For this example, our algorithm achieves 19% improvement as compared with the initial STSP solution.

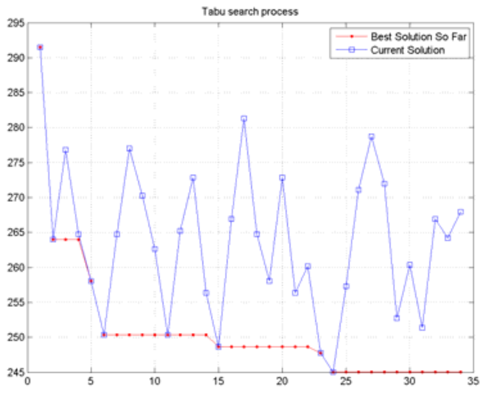

Figure.7 Tabu-search processes in finding the optimal solution, the search process stopped when the change of ten successive best so far solution is less than 0.1%.

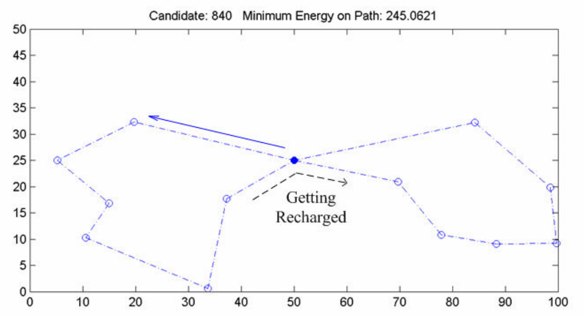

Figure.8 Tabu-search result.

V. Simulations And Results

In this section, we report the experimental results. We implement our algorithms in matlab, and test them on a Pentium PC (2.4GHz) with Windows XP and 1.5GB RAM. The results show that our algorithms achieve significant energy saving.

In the experiments, we construct benchmark programs with general consideration before subsequent experiment on mobile robot. We assume the maximum energy that a robot can have is 1000 energy units, and the robot consumes one unit of energy for traveling 1 unit distance. The energy of each task is randomly generated, and the value is in the range of 50~150 units.

In our experiments, we generate 8 groups of tasks (the number of tasks varies from 5 to 40) in 200*100 units' area. The energy required for finishing battery changing or recharging at the docking station is 50 units.

We compare the energy consumption of four algorithms, the TSP-Greedy-Based algorithm (GTSP), the sequential staying-alive method (STSP), the Tabu-Search-Based algorithm with the initial solution from GTSP (TS-GTSP) and the Tabu-Search-based algorithm with the initial solution from the sequential staying-alive method (TS-STSP). The experimental results are shown in Figure 9 and Table 1.

From the results, we can see that the quality of the tabu search method is good. For the instances with 10 tasks TS-STSP achieves 19.18% reduction as compared with GTSP method. For instances with 40 tasks TS-STSP contributes to 16.25% reduction. On average, our TS- GTSP and TS-STSP algorithms achieve 14.10% and 15.45% reduction on the energy consumption, respectively.

When the number of tasks is small, the results are almost the same among the four algorithms. The reason is that the path planning is relatively simple, and usually the robot has enough energy to directly go back to the docking station. With the increasing task number, we can see TS-STSP becomes better with its ability to deal with the complicated cases.

For the number of tasks equals to 30 and 35, we find that STSP consumes more energy than GTSP method. TS-STSP is more energy efficient than TS-GTSP. The reason is that sequential based path planning method consumes much energy in the inner of the sub route, seen in figure 6. Thus, using tabu-search method, energy can be greatly reduced.

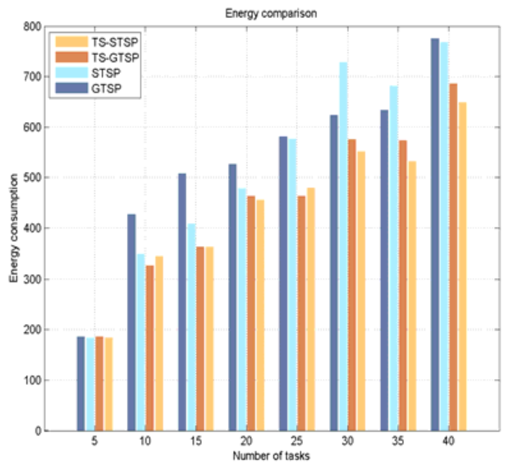

Figure.9 Energy comparison for four algorithms.

Nub of Tasks	GTSP	TS-GTSP	TS-GTSP Reduction (%)	TS-STSP	TS-STSP Reduction (%)
5	186.16	186.16	0	183.69	1.33
10	427.03	326.74	23.48	345.11	19.18
15	508.10	363.92	28.38	364.28	28.30
20	526.06	463.55	11.88	455.62	13.39
25	581.23	463.42	20.27	479.95	17.43
30	624.12	575.44	7.79	551.55	11.63
35	633.39	573.07	9.52	531.63	16.07
40	774.99	685.63	11.53	649.04	16.25

Table.1 Energy consumption comparison of TS-GTSP with GTSP and TS-STSP with GTSP

VI. CONCLUSION

To minimize energy consumption and keep mobile robots to stay alive becomes an important problem as most mobile robots are powered by batteries. In this paper, we proposed a dynamic energy-evaluation scheme to solve this problem. Combining our dynamic scheme with path planning, we proposed two staying-alive and energy-efficient path planning approaches based on the greedy TSP and Tabu-search methods, respectively. The experimental results show that our Tabu-search-based approach can provide an effective path planning and achieve significant energy saving.

VII. ACKNOWLEDGMENT

The work described in this paper is partially supported by the grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (PolyU A-PH13, PolyU A-PA5X, PolyU A-PH41, and PolyU BQ06B), the National Nature Science Foundation of China (60525314), the 973 Program of China (2002CB312204-04) and the 863 Program of China (2007AA041701).

REFERENCES

[1] A. Davids. "Urban Search and Rescue Robots: From Tragedy to Technology," IEEE Intelligent Systems, vol.17, pp. 81-83, March 2002.
[2] A. Drenner, I. Burt, T. Dahlin. "Mobility Enhancements to the Scout Robot Platform," In International Conference on Robotics and Automation, pp. 1069-1074, 2002.
[3] G.B. Danhig, J.H. Ramser, "The wck dispatching problem," Monagemenf Sdence, vol. 6, no. SO, 1959.
[4] M.Saha, G.Sánchez-Ante, "Planning multi-goal tours for robot arms," International Conference on Robotics and Automation, pp. 3797-3803, 2003.
[5] Y Hada, S.Yuta, "Robust navigation and battery re-charging system for long term activity of autonomous mobile robot," Intemational Conference on Advanced Robotics, pp. 297-302, 1999.
[6] M.C. Silverman, D. Nies, B. Jung, and G.S. Sukhatme, "Staying alive: a docking station for autonomous robot recharging," IEEE International Conference on Robotics and Automation, vol.1, pp. 1050-1055, 2002.
[7] S.Oh, A.Zelinsky and K.Taylor, "Autonomous Battery Recharging for Indoor Mobile Robots", Proceedings of Australian Conference on Robotics and Automation, 2000.
[8] Chong Hui Kim, Byung Kook Kim, "Energy-Saving 3-Step Velocity Control Algorithm for Battery-Powered Wheeled Mobile Robots," IEEE International Conference on Robotics and Automation, 2005
[9] R. Katoh, O. Ichiyama, T. Yamamoto, and F. Ohkawa. "A Realtime Path Planning of Space Manipulator Saving Consumed Energy," In International Conference on Industrial Electronics, Control and Instrumentation, pp. 1064-1067, 1994.
[10] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. S. G. Lee, "Energy-Efficient Motion Planning for Mobile Robots," International Conference on Robotics and Automation, pp. 4344-4349, 2004.
[11] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. S. G. Lee, "A case study of mobile robot's energy consumption and conservation techniques," International Conference on Advanced Robotics, pp. 492-497, 2005.
[12] A. Barili, M. Ceresa, and C. Parisi, "Energy-Saving Motion Control for An Autonomous Mobile Robot," In International Symposium on Industrial Electronics, pp. 674-676, 1995.
[13] J. Hurink and S. Knust, "A tabu search algorithm for scheduling a single robot in a job-shop environment," Discrete Applied Mathematics, vol. 119, pp.181-203, 2002.
[14] R.Aylett, "Robots: Bringing Intelligent Machines to Life?" Barron's Educational Series, New York, 2002.
[15] M. Jia, G. Zhou, and Z. Chen. "An Efficient Strategy Integrating Grid and Topological Information For Robot Exploration," IEEE Conference on Robotics, Automation and Mechatronics, pp. 667-672, 2004.
[16] F, Glover, Tabu Search, Part I and II, ORSA Journal on Computing vol.1, pp. 190-206, ORSA Journal on Computing, vol.2, pp. 4-32, 1990.
[17] M. Gendreau, A. Hertz and G. Laporte, "A tabu search heuristic for the vehicle routing problem," Management Science, vol. 40 (10): pp. 1276-1289, 1994.
[18] P.Zebrowski, R.T. Vaughan, "Recharging Robot Teams: A Tanker Approach," International Conference on Advanced Robotics, 2005.
[19] P.Zebrowski, Y.Litus, R.T. Vaughan, "Energy Efficient Robot Rendezvous," Canadian Conference on Computer and Robot Vision , pp.139-148, 2007.
[20] Floreano, D. and Mondada, F. "Evolution of homing navigation in a real mobile robot," IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26:396--407, 1996.
[21] E. W. Dijkstra, "A note on two problems in connexion with graphs," Numerische Mathematik, vol.1, pp. 269-271, 1959
[22] T.H. Cormen, C.E. Leiserson, Introduction to Algorithms, The MIT Press; 2nd edition, 2001.