Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2014) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014)

On the Adequacy of Tabu Search for Global Robot Path Planning Problem in Grid Environments

Imen Châari[a,b,*], Anis Koubâa[b,c,d], Hachemi Bennaceur[e], Adel Ammar[e], Sahar Trigui[a,b], Mohamed Tounsi[c], Elhadi Shakshuki[f], Habib Youssef[g]

[a] *University of Manouba ENSI, Manouba, Tunisia.*
[b] *Cooperative Intelligent Networked Systems (COINS) Research Group, Riadh, Saudi Arabia.*
[c] *Prince Sultan University, College of Computer and Information Sciences, Riadh, Saudi Arabia.*
[d] *CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Porto, Portugal.*
[e] *Al-Imam Mohamed bin Saud University, Research Unit of Sciences and Technology, Riadh, Saudi Arabia.*
[f] *Acadia University, Jodrey School of Computer Science, Canada.*
[g] *University of Sousse, PRINCE Research Unit, Sousse, Tunisia.*

Abstract

This paper investigates the capabilities of tabu search for solving the global path planning problem in grid maps. Accordingly, a tabu search system model is designed and a tabu search planner algorithm for solving the path planning problem is proposed. A comprehensive simulation study is conducted using the proposed model and algorithm, in terms of solution quality and execution time. A comparison between our results with those of A* and genetic algorithms (GA) is presented for small, medium and large-scale grid maps. Simulation results show that the tabu search planner is able to find the optimal solution for small scale environments. However, for large scale maps, it provides near-optimal solutions with small gap while ensuring shorter execution times as compared to the A* Algorithm. A discussion about the advantages and limitations of TS for solving a path planning problem is also presented.

Selection and peer-review under responsibility of Elhadi M. Shakshuki.

Keywords: Mobile robots; Global path planning; Tabu search.

1. Introduction

The problem of mobile robot path planning has received much attention in the robotic community. Indeed, in map-based navigation, the mobile robot is autonomous and is able to generate collision free paths to move from one location to another in its environment [1]. This problem is referred to as global path planning and is typically formulated as follows: *given a mobile robot and a model of the environment, find the optimal path between a start position and a final position without colliding with obstacles*. The optimality of the path depends on the optimization criteria such

* Imen Châari. Tel.: +33-695-942-657.
E-mail address: imen.chaari@coins-lab.org

1877-0509
Selection and peer-review under responsibility of Elhadi M. Shakshuki.

as shortest length, minimum time or energy, and/or cost.
In the literature, the path planning problem is categorized by two factors: (1) the environment which can be static or dynamic and (2) the robots knowledge about the environment; if the robot has a complete knowledge about the environment, this problem is known as global path planning. On the contrary, if the robot has a partial knowledge, this problem is classified as local path planning [1]. In this paper, we consider the problem of global path planning in static grid environments. Several heuristic solutions and approaches have been proposed to solve the global path planning problem such as Ant Colony Optimization (ACO) [2], Genetic Algorithms (GA) [3], Particle Swarm Optimization (PSO) [4], and Tabu Search (TS) [5]. Other techniques are based on exact methods, which have the advantage of being complete and guarantee finding the optimal solution if it exists. These diversity and difference raise the complex challenge of choosing the best algorithm for solving the path planning problem. We are addressing this research question in the iroboapp project [6], aiming at understanding the capabilities and performance of existing approaches for solving the global path planning problem and design new hybrid algorithms for efficient path search in large-scale environments. It is under this context that we are investigating, in this paper, the capabilities of the tabu search approach. Indeed, tabu search approach is effective in solving several optimization problems, but relatively unexplored for the global path planning problem in grid environments. Looking at the the literature, there exist several attempts to compare or combine the tabu search approach with other techniques, such as genetic algorithms and simulated annealing [7]. It should be noted that, there is no major research efforts that presented a comprehensive solution on how to apply tabu search for mobile robot global path planning in grid environments. This represents the main motivation of the work presented in this paper. This paper aims at filling the gap and presenting the following two main contributions: (1) the design of a new tabu search approach for solving the global path planning problem in grid environments. We carefully adapted and applied the different theoretical concepts of tabu search and designed suitable moves and neighborhood for increasing search efficiency. (2) Evaluation of capabilities and performance of the tabu search approach for small, medium and large-scale environments. Our main objective is to understand the advantages and limitations of tabu search for solving the global path planning problem.
The rest of this paper is organized as follows. In Section 2, we describe the tabu search approach. Section 3 presents a detailed description of the tabu search algorithm (TS-PATH). The simulation results are presented in Section 4. An overview of the works that addressed the robot global path planning using tabu search is presented in Section 5. Section 6 concludes the paper and discusses future works.

2. Tabu Search Concepts for Global Path Planning in Grid Environments

This section discusses the basic principles and concepts pertaining to applying tabu search for solving the global path planning problem in grid environments. Tabu search is one of the problem solving techniques to combinatorial optimization problems introduced by Fred Glover [8], [9]. It is a local search method that starts from an initial solution, which can be randomly generated or computed using a greedy algorithm, then iteratively attempts to improve the current solution around an appropriately defined neighborhood. The transition between the current solution and another solution in the neighborhood is called a move. Contrary to the greedy search method, which stops the search when no better solution can be found in the neighborhood, the tabu search approach pursues the search whenever a local optimum is encountered by allowing non-improving moves. The search process stops when a predefined termination criterion is satisfied.
Tabu search has been applied with great success to a large variety of difficult combinatorial optimization problem areas [10], such as assignment [11], routing [12], the Travelling Salesman Problem [13], etc. However, this technique is relatively unexplored for the path planning problem in particular for grid environments.
Designing and applying the tabu search technique to the robot path planning problem requires answering and mastering the following important questions, which we answer in this paper: How a solution of the path planning problem could be encoded? How the first feasible solution is generated? What is the neighborhood structure of a solution? How to compute the costs of solutions neighbors? How to manage the memory ? How to fix the parameters of the Tabu search as stopping condition, the size of neighborhoods, etc?

The Environment. A grid-based model represents the navigation area of the mobile robot.The grid map is divided into equal size cells. Each cell is given a unique number, starting from 0 for the top left cell, 1 for the next cell to the right

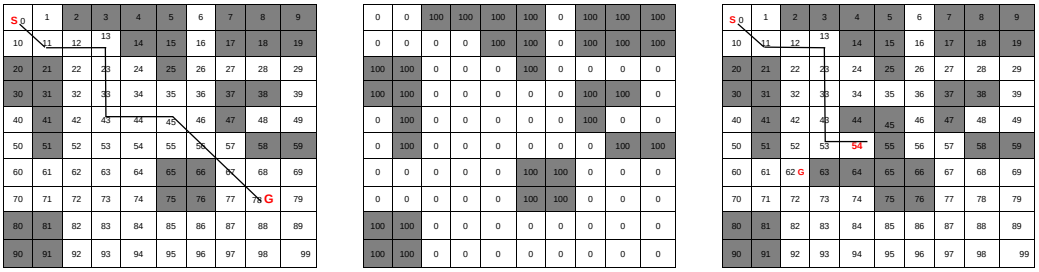

Fig. 1. A 10 x 10 Grid Environment

and so on. Fig. 1. a depicts an example of 10 x 10 grid map. One main reason that led us to consider grid map is that we would like to integrate our path planner into the Robot Operating System (ROS) framework [14] for robot application development. Based on ROS convention, the grid map is represented by two dimensional matrix called occupancy grid map. Each cell of the matrix contains an occupancy information of the part of the environment that it covers. The occupancy information of a cell is represented by a unique number that can have one of these three possible values 0, 100 or -1. Where 0 means the cell is free, 100 means the cell is occupied by an obstacle, and -1 means the cell is unknown. An unknown cell is considered as an obstacle in the path planning process. Fig. 1. b depicts an example of a 10 x 10 occupancy grid map. We assume that the robot can move in horizontal, vertical and diagonal directions from a cell to another (8 adjacent grid cells at most). The obstacles are static and known in advance.

Solution encoding. The robots path is encoded by a sequence of free cells starting from the start cell to the goal cell. Fig. 1. a shows an example of a feasible path. . In this figure, the start cell is 0 and the goal cell is 78. The path is encoded as (0, 11, 12, 13, 23, 33, 43, 44, 45, 56, 67, 78). The path cost is the sum of the action sequence of the robot. The vertical and horizontal moves in the grid are assigned a 1 unit distance; whereas, the diagonal moves are assigned a 1.4 unit distance. The path cost for the proposed is calculated as: (1.4+1+1+1+1+1+1+1+1+1.4+1.4+1.4) = 13.6. It should be noted that the generated paths during the search process may have variable sizes (variable path length).

Solution neighborhood and moves. As a local search algorithm, the tabu search method tries iteratively to improve the current solution around its neighborhood until a predefined termination criterion is satisfied. Each neighbor solution is reached from the current solution by applying a small transformation called move. For the global path planning problem, a move consists of applying one transformation to a feasible path in order to obtain a new feasible path. There are three possible basic moves that can be considered in the global path planning in grid environment: 1) insert move, 2) remove move, and 3) exchange move. Fig. 2 illustrates three examples of the three possible moves. The first example shows inserting cell 13 between cell 12 and cell 23 of Path 1. The second example shows removing cell 22 from path 2. The last example shows cell 11 is replaced by cell 1. With respect to the map in Fig. 1.c, all the new paths are feasible.

The move may either improve the current solution (i.e. decreases the path cost) or deteriorate it (i.e. increases the path cost) in terms of path length. Typically, an insert move would increase the path length, but in tabu search it is permissible to consider solutions with lower quality to investigate its neighborhood and look for possible better solutions after certain iterations. A remove move, however, would generally decrease the path length. Additionally, it leads to a better solution if the resulting path is feasible. During the search process, it is necessary to verify that the applied moves produce a feasible path. In addition, before accepting a move to generate a new path two conditions must be verified: 1) The move must improve the current path, i.e. the new path cost is smaller or at least equal to that of the current path, and 2) the move is not tabu, which means that the move doesnt exist in the tabu list. This will be explained in more detail in the following section. The neighborhood of a given path may be too large and consequently the evaluation of the cost of all neighbors will be very time consuming. To overcome this problem, the following solutions are proposed: *(1). Best among all.* It scans all the paths found in the neighborhood of the

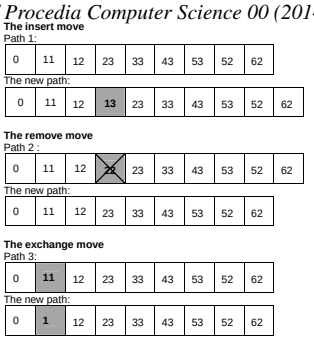

Fig. 2. Insert, Remove and Exchange Moves

current path and then it selects the best neighboring path that has the lower cost. The problem of this method is time-consuming because of the generation and the evaluation of all candidates in the neighborhood. *(2). First-improving.* It selects the first path found in the neighborhood that has a cost smaller than the current path. This method has the advantage of being fast, because it stops the search of neighborhood once a new better path is found. *(3). Best among first C (C random).* First a set of C neighboring paths of the current path is generated where C is random number. Then, the path that has lower cost than the current path is selected. *(4). Best among first C (C constant).* First, C neighboring paths of the current path are generated where C is constant. Then, the path that has a lower cost than the current path is selected.

Tabu list. To avoid being trapped at a local optimum and backtracking to already visited paths, the tabu search approach keeps track of the recent solutions in a temporary buffer referred to as *Tabu List*.
For the case of global path planning in grid environment, the content of the Tabu List can be defined according to two possible strategies [10]. The first strategy consists in storing all recently visited paths in the Tabu List. This strategy is not scalable as it increases the space and time complexity and the algorithm will take a long time to check the contents of the list to prevent revisiting already visited paths. On the other hand, the second strategy consists in storing only the moves that make the transition from one solution to another. This strategy has the advantage of being fast as compared to the first strategy as we have to check only the existing tabu moves before performing a move; also it consumes less space. Thus, in our work we adopted the second strategy. Tabu list is a powerful mechanism that prevents revisiting recently visited moves. However, it may prohibit some attractive moves which improve the current path cost. Thus it is necessary to allow revoking the status tabu of these moves in order to use them in the search process. This mechanism is known as aspiration criteria.

3. TS-PATH: The Tabu Search Planner

In this section, we describe the tabu search planner, i.e. TS-PATH. The pseudo-code of the algorithm is presented in Algorithm. 1.
Step 1: Generation of the initial solution (Line 1 in Algorithm 1): The first step of TS-PATH consists of generating an initial path from the start position to the goal position. The initial solution is constructed using the greedy method based on the euclidian distance heuristic. The initial path must be feasible (all the adjacents cells in the path are connected and do not contain obstacles).
During the construction of the initial path, it is possible to fall into a deadlock situation, meaning that the last cell in the path is surrounded by obstacles or already visited cells. This means that there are no more choices to select for the next cell in the path. In such situation, backtracking operation needs to be applied to recover from the deadlock position and move back to an earlier position with free unvisited neighbors. Fig. 1.c illustrates an example of a deadlock situation, the start position is 0, the goal position is 62. The robot reaches cell 54 which is surrounded by 6 obstacles (44, 45, 55, 63, 64 and 65) and 2 visited cells (43 and 53). Thus, the robot should move back to cell 53 and choose one of its unvisited neighbors cells different from 54. In this case, there are three possible cells (42, 52 and 62). The initial path generated by the greedy method is then divided into sub-paths, called segments. The segment size for a given path is determined experimentally, and we call this process *segmentation*. Each segment is considered a path,

Algorithm 1. The TS-PATH Algorithm

```
1:  Generate the initial feasible path using the greedy algorithm
2:  Current path= initial path
3:  repeat
4:    for each cell in the current path do
5:      if Move(current cell) is not tabu then
6:        Generate the new path after applying the move: exchanging, inserting, removing the current cell
7:        Calculate the new path cost
8:        if new path cost ≤ current path cost then
9:          Make the move tabu
10:         Current path= new path
11:         Add the new path to the set of candidate best paths
12:       end if
13:     else
14:       if the move is tabu and new path cost ≤ current path cost (aspiration criteria) then
15:         Add the new path to the set of candidate best paths
16:       end if
17:     end if
18:   end for
19:   Update the tabu lists
20:   Add the best path from the set of candidate best paths generated in one iteration
21: Until { generation number ≤ maximum generation number }
22: Select the robot's path
23: Generate a new initial path by applying diversification() (Algorithm 2)
24: go to 3 (apply the tabu search algorithm on the new initial path)
```

the tabu search algorithm is applied on each segment to generate new segments that improve its cost. At the end of an iteration, the best segments are recombined to constitute the best path. The segmentation contributes to reducing the search time, which was confirmed by simuations.

Step 2: Generation of the neighborhood of the current solution (From Line 5 to line 17 in Algorithm 1):

The Definition of the Neighborhood Structure: The constructed initial solution represents the entry point for the TS-PATH algorithm. For each iteration, applying moves to the current solution creates the neighbourhood. We devised three possible moves delete, insert and replace, which are presented in Section 2. Given any path p of length L from the initial state S to the goal state G, the neighborhood of p is defined by performing the following moves:

- Cell removing: for each cell of p different from S and G check the possibility to remove it in order to obtain another feasible path $p\prime$.
- Cell exchanging: for each cell x of p different from S and G check the possibility to replace it by another cell y not in p in order to build another feasible path $p\prime$.
- Cell insertion: for each cell x not in p check the possibility of adding it to p in order to build another feasible path $p\prime$.

Carrying out cell removing and exchanging may improve the cost of the current best path but unfortunately cell insertion move always degrades the current best path. The cell insertion move may be useful for exploring new region around local optimums. Notice that the size of neighborhood is $O(L)$ where L is the length of the current best path.

Tabu List , Tenure: Two Tabu Lists are used in the algorithm: TabuListIn and TabuListOut. They are two lists of maximum size 2 x Tenure. TabuListIn contains the arcs (cell i, cell j) that are added after carrying out a move and TabuListOut contains the arcs (cell i, cell j) which are removed after performing a move. Each move saved in the list is

characterized by four attributes fromCell, ToCell, Tenure and ExpirationDate. Considering the following example, if two cells are swapped, cell 11 and cell 21. In tabuListIn we added the arcs (10,21) and (21,20) and in tabuListOut we added the arcs (10,11) and (11,20). For the arc (10,21): fromCell=10, toCell=21, Tenure=k, ExpirationDate=current number of generation+Tenure. This is to prevent the algorithm from going back on this move during (number of generation+Tenure) iterations. Once an arc is stored in the tabu list, the tabu status tabu is assigned to it. As indicated above, the maximum size of one list is 2 x Tenure; Indeed, the maximum number of arcs that will be added in the TabuListIn or in TabuListOut after carrying out one of the aforementioned moves is 2. Moreover, each move in the tabu List stays tabu for tenure iterations. So, the maximum size of one list will not exceed 2*tenure. As mentioned in the previous section, before accepting a move to generate a new path we must verify that the move is not tabu, we must check that the arcs that will be added do not exist in TabuListIn and the arcs that will be removed do not exist in the TabuListOut. After each iteration of the algorithm, the two tabu lists are updated. If the expirationDate of one move is equal to the current iteration number the arc is removed from the tabu list.

Aspiration Criteria: During the generation of the current path neighborhood, some moves might be accepted even if they are tabu. These moves have the capability to improve the current best solution (their costs are inferior to the current path's cost). So, it is useless to keep them in the tabu list and it is preferable to liberate them. It is a certain manner to relax the mechanism of the tabu list. In other words if a given tabu move in the tabu list can improve the current best solution found so far (this may happen) then it will be removed from the tabu list and evaluated as all other allowed moves. Moreover in this case we are sure to improve the current best solution and so we are sure that when performing this move there is no cycle (we do not go back to explored solutions). Therefore, there is no need to keep this move in the tabu list.

Step 3: Incremental cost evaluation of neighbor paths:

The cost of each neighbour must be quickly evaluated. We adopted an incremental computing of the following new paths costs as follows: Let p be a feasible path neighbour of the path p, generated by the three moves (insert, remove, exchange).

- If p is obtained by a cell insertion move of a cell x between two cells $x1$ and $x2$ of p, then $cost(p)$=$cost(p)$-$cost(x1, x2)$ + $cost(x1, x)$ + $cost(x, x2)$
- If p is obtained by a cell removing move of a cell x from p between $x1$ and $x2$, then $cost(p)$=$cost(p)$+$cost(x1, x2)$-$cost(x1, x)$-$cost(x, x2)$.
- If p is obtained by a cell exchanging move of cell x in p between cells $x1$ and $x2$ and cell y not in p, then $cost(p)$=$cost(p)$-$cost(x1, x)$-$cost(x, x2)$+$cost(x1, y)$+$cost(y, x2)$.

The incremental computing of the costs leads to save time and permits to explore large neighbourhoods.

Step 4: Diversification:

When the search is stagnated during a certain number of consecutive iterations, diversification is used to drive the search towards a new region of the search space. The diversification method begins with drawing a straight line between the start and the goal positions using the greedy method based on the Euclidean distance heuristic, this line could intersect with obstacles. At the radius of N cells (N is a random parameter), the algorithm choose a random intermediate cell, which will be used to generate .a new feasible path from the start to the goal cells across it. The new generated path will be used as an initial solution to restart the tabu search. The fact that the new path involves an intermediate cell generated randomly gives a chance to explore different region than those explored so far.

4. Performance Evaluation

4.1. Simulation model

To perform the simulation, we designed an object-oriented simulation model and implemented it using C++ under Linux OS. All simulations are implemented on a PC with an Intel Core i7 CPU @ 2.40 GHz and 8GB of RAM. We tested the algorithm on different-size maps with different complexities [15](i.e. obstacle ratio) and randomly chosen start and goal positions. The grid maps sizes vary from (10 x 10) up to (2000 x 2000) cells. To impose different environment complexities, we varied the obstacle ratio from 0.1 to 0.3 for every map.

4.2. Simulation results

4.2.1. Performance evaluation of the tabu search planner

In this section, we present an extensive simulation study to evaluate the efficiency of the tabu search algorithm. The objective of the simulation is two-folded: First, we evaluate the planner performance in terms of path quality and execution time by comparing it against A*. Second, we compare the path planner against GA algorithm presented in[3]. For that, we consider three different groups of maps: small grid maps, medium grid maps and large grid maps. Each group of maps contains different map sizes. Small grid maps includes maps size from 10 x 10 up to 30 x 30, medium grid maps includes 50 x 50, 60 x 60 and 100 x 100 grid maps and the last group includes 500 x 500, 1000 x 1000 and 2000 x 2000 grid maps.

Optimality and convergence time: For ensuring a reliable statistical analysis, we considered 30 different scenarios for each map size. Where each scenario is specified by the coordinates of a randomly choosen start and goal cells. Each scenario, with specified start/goal cells, is repeated 30 times (i.e. 30 runs for each scenario). The average values of the metrics are then calculated with 95% of confidence interval. In total, 900 runs for each map size are performed in the performance evaluation study.
Two metrics are considered to evaluate the algorithm: (1). The *Relative Gap*=$\frac{Cost(solution\,found)}{Cost(optimal\,solution)}$ between the optimal solution found by the A* algorithm and the best solution found by the tabu search algorithm; (2). The *Relative Time*=$\frac{Time(solution\,found)}{Time(optimal\,solution)}$ which is defined as the ratio of the time to find the best solution by the tabu search algorithm to the time to find the optimal solution.

We note that the time to find the best solution by the tabu search algorithm includes the time to find the initial solution using the greedy method plus the tabu search processing time.

Fig. 3, Fig. 4 and Fig. 5 present the scatterplot of the Relative Time versus the Relative GAP for different map sizes. In fact, we observe for small scale maps that the most of the scattered points are concentrated around the y-axis (the most of the Relative Gap values are equal to 0) which means that the tabu search algorithm is able to find the optimal solution found by the A* algorithm for the most of cases as it is illustrated in Fig. 3. Looking at Fig. 4 and Fig. 5, we notice that the tabu search approach fails to find the optimal solution found by A* for medium and large grid maps. The average Relative Gap and the average Relative Time calculated over the 30 scenarios for each map size with 95% of confidence interval are presented in Table 1. We can see that the most of the non-optimal solutions have a small GAP.

Table 1. Average Relative GAP and Average Relative Time for the Different Map Sizes

Map size	Average Relative GAP	Average Relative Time
10*10	1.008 ± 0.010	0.9 ± 0.401
20*20	1.017 ± 0.015	1.154 ± 1.057
30*30	1.039 ± 0.037	0.766 ± 0.31
50*50	1.03 ± 0.013	1.481 ± 0.694
60*60	1.022 ± 0.099	3.159 ± 0.81
100*100	1.085 ± 0.0301	1.45 ± 0.7658
500*500	1.089 ± 0.025	0.324 ± 0.1565

From Fig. 3 and Fig. 4 we notice that for small and medium grid maps, the tabu search algorithm exhibits longer execution times than A* to find its best solution. However, for large grid-maps, we observe that the scattered points are concentrated around the x-axis which means that the tabu search algorithm finds its best solution much faster than A*. We conclude that the tabu search approach can be used in large-scale grid maps to find good (but non optimal) solutions with small gap much faster than A* (especially for far away start/goal cells). However, for small and medium grid maps A* can be used as it always exhibits the best solution qualities and shortest execution times.

Comparison against GA: To compare the two path planners, three performance metrics are assessed: *(1) the path cost*: it represents the cost of the shortest path found by an algorithm, *(2) the execution time*: it is the time spent by an algorithm to find its best (or optimal) solution. Table 2, Table 3 and Table 4 present the path costs and the execution times of tabu search, GA and A* for different map sizes with different obstacle ratios. We notice from Table 2, that both planners GA and tabu search provide the optimal paths as compared to the A* algorithm for all the map sizes

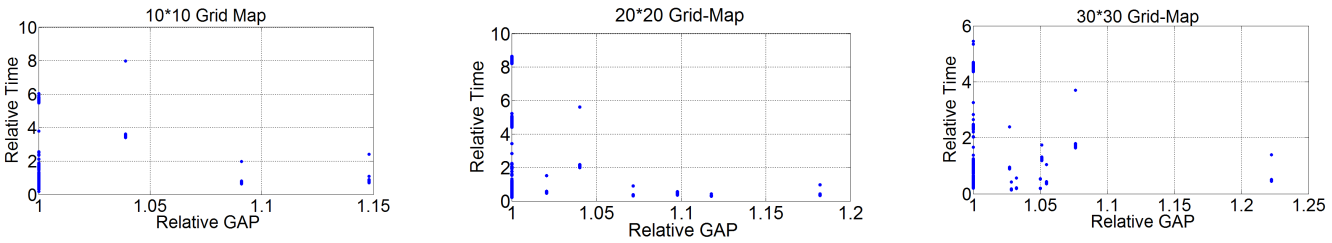

Fig. 3. Scatter Plot of the Relative Time versus the Relative GAP for Small Grid-Maps

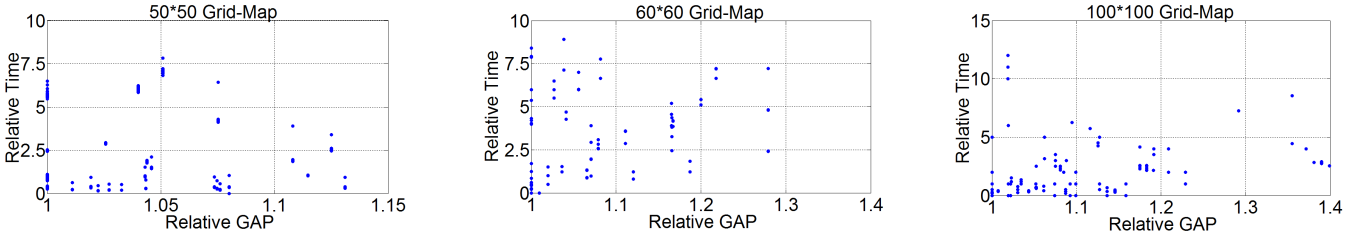

Fig. 4. Scatter Plot of the Relative Time versus the Relative GAP for Medium Grid-Maps

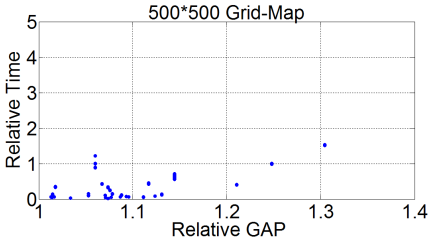

Fig. 5. Scatter Plot of the Relative Time versus the Relative GAP for large Grid-Maps

with different obstacle ratios. For medium and large scale maps, we can conclude that tabu search always falls into a local optimum and fails to find the optimal path. However, GA is able to find the optimal solution provided by A* for near start/goal cells as it is depicted in Table 3 and Table 4. For far start/goal cells, GA also fails to find the optimal solution as it is the case for the scenario (0, 249999) in 500 x 500 grid map. This does not necessarily mean that A* outperforms tabu search for large scale maps because it always needs longer time to converge to the optimal solution as it is illustrated in Table 4. The GA algorithm always provides slower convergence speed as compared to the two other algorithms. This can be explained by the fact that the tabu search is an optimization approach that rely on the initial path generated by the greedy method which is not necessarily the path close to the best path. Especially, in large environments, after a certain number of iterations the quality of the path is improved but it cannot reach the best solution by applying simple moves (removing cells, exchanging cells). For that reason, we only used tabu search for post optimization purposes to improve on other optimization approaches solution quality.

5. Related Works

Although the effectiveness of tabu search to deal with combinatorial problems is proved, a few of research works based on this technique were proposed for solving the path planning problem. In[5] and[16], the authors claimed to be the first to propose a full TS-based solution for the local path planning problem in static grid-map environments, where they showed TS is effective. The idea consists in finding a set of Tabu moves in each iteration of the search

Table 2. Results of Comparison of A*, GA and Tabu Search for Small Grid Maps (Path Cost/Execution Time (ms))

Map	Obstacle Ratio	Obstacle Size	Start Cell	Goal Cell	A*	GA	TS initial path	TS best path
	0.1	1	10	98	14.4 / 1.86922	14.4 / 36.989	23.2 / 1.80083	14.4 / 17.1067
10*10	0.2	2	10	98	14.4 / 1.71379	14.4 / 37.449	20.6 / 1.52501	14.4 / 19.2276
	0.5	2	80	4	9.6 / 0.476017	9.6 / 19.2938	10.4 / 0.52198	9.6 / 1.6242
	0.1	1	200	0	10.8 / 0.53828	10.8 / 30.975	11.6 / 0.65704	10.8 / 2.32407
20*20	0.2	2	365	227	7.8 / 0.19987	7.8 / 17.325	8.6 / 0.173698	7.8 / 0.955188
	0.3	3	47	200	18.2/1.97313	18.2 / 191.709	48.8 /6.87727	18.2/ 115.205
	0.1	1	200	769	20.2 / 2.5209	20.2 / 54.820	21 / 2.03291	20.2 / 24.2811
30*30	0.2	2	826	387	20 / 1.8526	20/ 51.0076	25.6 / 1.95537	20 / 29.6026
	0.3	3	200	850	32.6 / 8.8029	32.6 / 555.62	48.2 / 8.57062	32.6 / 41.1076

Table 3. Results of Comparison of A*, GA and Tabu Search for Small Grid Maps (Path Cost/Execution Time (ms))

Map	Obstacle Ratio	Obstacle Size	Start Cell	Goal Cell	A*	GA	TS initial path	TS best path
	0.1	1	50	2345	64 / 20	66,6/270	70 / 10	64,4 / 40
50*50	0.2	5	2423	150	54.2 / 20	54.2 / 0	55 / 0	54.2 / 10
	0.3	3	0	2100	48.4 / 10	48.4 / 1830	52 / 10	49.6 / 30
	0.1	3	30	3599	71.8 / 80	72.6 / 1590	78.2 / 0	73.4 / 1440
60*60	0.2	5	100	2566	45 / 20	45 / 4490	59.6 / 10	45 / 1090
	0.3	5	20	3420	70.4 / 70	70.4 / 2480	154.8 / 30	88.4 / 1470
	0.2	2	10	5968	84 / 20	84 / 9280	102.6 / 30	95 / 130
	0.3	2	10	6000	65.6 / 30	67.2 / 5770	108 / 40	71 / 420
100*100	0.4	2	10	9000	111.2 / 90	114.6 / 103990	248.2 / 100	146.2 / 1100
	0.4	5	0	9999	152.4 / 220	152.4 / 25040	461.399 / 290	249.8 / 2870
	0.25	10	0	9999	147 / 160	147 / 37180	189.8 / 40	162.8 / 1420
	0.370	2	400	9988	147.6 / 360	156.8 / 185980	275.4 / 90	191.8 / 1570

Table 4. Results of Comparison of A*, GA and Tabu Search for Large Grid Maps (Path Cost/Execution Time (ms))

Map	Obstacle Ratio	Obstacle Size	Start Cell	Goal Cell	A*	GA	TS initial path	TS best path
	0.1	5	0	249999	716.002/47060	716.602/100000	737.802/600	719.602/4270
	0.2	5	0	249999	728.602/100040	735.802/2593530	812.803/750	767.602/6010
500*500	0.2	20	0	249999	741.802/302080	766.802/206000	1018.01/125	770/20110
	0.3	5	0	249999	750.802/210430	767.602/20146660	886.203/920	800.4/11360
	0.2	50	202362	202567	317/5190	317/170980	359.2/220	358.6/4270
	0.19223	20	200000	558440	604.799/110400	604.799/442140	654.999/500	625.7/119710
1000*1000	0.18584	50	355555	999999	821.601/401810	821.601/474640	1003.6/1210	696.401/6420
	0.27745	20	100000	558440	665.801/47170	691.001/113650	705.001/620	696.401/6420
2000*2000	0.192491	50	903526	1641700	438.599/34720	438.599/113150	538/340	483.7/57530
	0.192772	100	164142	803526	461/29880	461/880000	552.8/430	551.6/5560

to confine the locations of the robot and guide its motion until reaching the destination. In[7], the authors conducted a comparative study between different metaheuristic approaches.Three methods were evaluated: tabu search, simulated annealing and genetic algorithms. The authors did not provide any information about the size of the neighborhood, the length of a solution (number of nodes included in a solution) and the way of evaluating the costs of neighbours. Moreover, the authors didnt compare their planners against an exact method. Simulated annealing was shown to outperform the other planners in terms of execution time, while tabu search was proved to provide the best solution in

terms of path length. The authors tested their algorithm in different environmentsand proved its effectiveness in terms of path length and execution time.

6. Conclusions and Future Works

In this paper, we studied the efficiency of the tabu search approach for the robot global path planning problem. The main contribution consists in the investigation of the adequacy of tabu search approach and understanding its advantages and limitations as a global path planner for mobile robots in grid environments. We designed a tabu search algorithm for path planning in grid environments and extensively evaluated its performance for different maps sizes (small, medium and large scales) and compared it against A* and GA. We conclude that TS can be effective for finding near optimal solutions with small gap in shorter times than A* in large-scale environments. We found out that the TS execution time is only 32% that of A* for finding solution with an average gap less than 10% for large-scale maps. For small size maps, TS is able to find the optimal solution but with an execution time greater than that of A*. It turns out that TS can be effectively used as a post-optimization process of constructive meta-heuristic methods such as Ant Colony Optimization (ACO) or Particle Swarm optimization (PSO) to improve their solutions. Currently, we are working towards designing hybrid path planners based on constructive meta-heuristic methods to optimize both solution quality and execution time.

Acknowledgements

This work is supported by the iroboapp project "Design and Analysis of Intelligent Algorithms for Robotic Problems and Applications"[6] under the grant of the National Plan for Sciences, Technology and Innovation (NPSTI), managed by the Science and Technology Unit of Al-Imam Mohamed bin Saud University and by King AbdulAziz Center for Science and Technology (KACST). This work is partially supported by Prince Sultan University.

References

1. P. Raja, S. Pugazhenthi, Optimal path planning of mobile robots: A review, International Journal of Physical Sciences 7 (9) (2012) 1314–1320.
2. I. Chaari, A. Koubaa, H. Bennaceur, S. Trigui, K. Al-Shalfan, smartpath: A hybrid aco-ga algorithm for robot path planning, in: In 2012 IEEE Congress on evolutionary Computation (CEC), Brisbane, Australia, 2012, pp. 1–8.
3. M. Alajlan, A. Koubaa, I. Chaari, H. Bennaceur, A. Ammar, Global path planning for mobile robots in large-scale grid environments using genetic algorithms, in: 2013 International Conference on Individual and Collective Behaviors in Robotics ICBR'2013, Sousse, Tunisia, 2013.
4. N. A. Shiltagh, L. D. Jalal, Optimal path planning for intelligent mobile robot navigation using modified particle swarm optimization, International Journal of Engineering and Advanced Technology (IJEAT) 2 (4) (2013) 260–267.
5. E. Masehian, M. R. Amin-Naseri, A tabu search-based approach for online motion planning, in: IEEE International Conference on Industrial Technology, Mumbai, India, 2006, pp. 2756–2761.
6. iroboapp: Design and analysis of intelligent algorithms for robotic problems and applications, http://www.iroboapp.org.
URL `http://www.iroboapp.org`
7. A. Hussein, H. Mostafa, M. Badrel-din, O. Sultan, Metaheuristic optimization approach to mobile robot path planning, in: International Conference on Engineering and Technology (ICET), Cairo, Egypt, 2012, pp. 1–6.
8. F. Glover, Tabu search - part i, ORSA Journal on Computing 1 (3) (1989) 90–206.
9. F. Glover, Tabu search - part ii, ORSA Journal on Computing 2 (1) (1990) 4–32.
10. F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, 1999.
11. T. V. Luong, L. Loukil, N. Melab, E. Talbi, A gpu-based iterated tabu search for solving the quadratic 3-dimensional assignment problem, in: 2010 IEEE/ACS International Conference on Computer Systems and Applications (AICCSA), Hammamet, Tunisia, 2010, pp. 1–8.
12. L. Bouhafs, A. Hajjamand, A. Koukam, A tabu search and ant colony system approach for the capacitated location-routing problem, in: Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD '08., Phuket, Thailand, 2008, pp. 46–50.
13. Y.-F. Lim, P.-Y. Hong, R. Ramli, R. Khalid, A tabu search and ant colony system approach for the capacitated location-routing problem., in: Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD '08., Phuket, Thailand, 2008, pp. 46–50.
14. M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. N. Ros, An open-source robot operating system, in: In ICRA Workshop on Open Source Software, Kobe, Japan, 2009.
15. Grid-maps: 10 x 10 up to 2000 x 2000.
URL `http://www.iroboapp.org/index.php?title=Maps`
16. E. Masehian, M. R. Amin-Naseri, Sensor-based robot motion planning - a tabu search approach, in: In IEEE Robotics and Automation Magazine, Vol. 15, 2008, pp. 48–57.