Sensor-Based Robot Motion Planning

A Tabu Search Approach

Planning robot motions in unknown environments has been an attractive research theme for many roboticists during the past two decades. The class of motion planners dealing with this kind of problem is known as online, sensor-based, local, a posteriori, real-time, or reactive motion planners.

Among the first works on online motion planning is Lumelsky's Bug algorithm presented for a point robot to move from a source point to a destination point, using touch sensing in a planar terrain populated with arbitrarily shaped obstacles [1]. Cox and Yap developed algorithms to navigate a rod to a destination position in planar polygonal terrains [2]. A survey on early online path planning works is provided in [3]. Another noteworthy approach for real-time planning is Khatib's potential fields (PFs) method, in which a point robot is directed by the forces in a field of potentials exerted by repulsive obstacles and the attractive goal [4].

Aiming to take advantage of the properties of roadmaps constructed usually in the offline mode, some researchers have tried to utilize the distance transform approach to build them incrementally. In [5], an algorithm is proposed for the navigation of a circular robot in unknown terrains by iteratively visiting the vertices of the Voronoi diagram. Choset developed an incremental method to construct the hierarchical generalized Voronoi graph (HGVG) [6], which exploits some bridge edges (called GVG^2) to maintain the connectivity of the GVG in high dimensions. Also, another method is proposed in [7] for online motion planning through incremental construction of medial axis.

©DIGITAL VISION

Other works such as [8] have tried to guide the motions of the robot along the edges of the visibility graph of a workspace of convex polygons in online mode. In [9], an algorithm is presented in which the visibility graph of the workspace is incrementally constructed by integrating the information of the paths traversed so far, and then, a globally optimal path is planned after the graph completion, as in offline mode.

In addition to the classic motion-planning approaches, other optimization methods generally known as heuristics have been increasingly employed for planning and optimizing robot motions. Heuristic algorithms do not guarantee to find a solution, but if they do, they are likely to do so much faster than the competing complete methods.

Some well-known metaheuristics such as genetic algorithms (GAs) and simulated annealing (SA) have found applications in robot motion planning. In [10], the path planning problem is expressed as an optimization problem and solved with a GA. It is done by building a path planner for a planar arm with 2 degrees of freedom, and then for a holonomic mobile robot. In [11], the path planning for vehicles is formulated as an optimization problem; the goal is to choose a path connecting initial and final points that crosses the least number of obstacles (with the eventual goal of zero crossings) in configuration space. A GA is devised in which the population is a set of paths. The SA approach is widely used in combination with the artificial potential field approach to escape from local minima, as in [12].

Digital Object Identifier 10.1109/MRA.2008.921543

BY ELLIPS MASEHIAN AND MOHAMMAD REZA AMIN-NASERI

The Tabu Search

The tabu search (TS) method is another well-known metaheuristic technique first introduced by Fred Glover in 1989 [13]. TS is a powerful algorithmic approach that has been applied with great success to a large variety of difficult combinatorial optimization problem areas, such as assignment, scheduling, routing, TSP, etc.

TS has three phases: preliminary search, intensification, and diversification. During the first of these three steps, TS is similar to some other optimization methods in that whatever point x in the input space the robot is currently at, it evaluates the criterion function $f(x)$ at all the neighbors N of x and finds the new point x' that is best in N. Repeating this idea creates the possibility of endlessly cycling back and forth between x and x' (a local minimum). To avoid this, TS differs from many other methods in that the robot moves to x' even if it is worse than x.

TS keeps track of performed moves and labels the recent moves as tabu moves, meaning that the search shall not revisit these points. The set of tabu moves is called tabu list (which is actually a push-down stack of s elements managed in a first-in/first-out manner), and its size is called tabu size. So that, for instance, once the move $x \to x'$ has been made, the reverse move $x' \to x$ is forbidden for at least the next s moves. Of course, when a tabu move has a cost lower than an aspiration level, it can be selected regardless of its tabuness. In addition to this tabu list, which is a recency-based short-term memory, we might introduce a frequency-based memory that operates on a much longer horizon (e.g., the last 50 iterations) and penalize the most frequently visited moves.

In the second (intensification) part of the search, it 1) starts with the best solution found so far (which is always stored throughout the entire algorithm), 2) clears the tabu list, and 3) proceeds as in the preliminary search for a specified number of moves. Finally, in the diversification phase, the tabu list is cleared again, and the s most frequent moves of the run so far are set to be tabu. Then, it chooses a random x to move to and proceeds as in the preliminary search phase for a specified number of iterations. The intensification phase focuses on the promising regions discovered during preliminary search, whereas the diversification phase forces exploration of completely new regions [14].

In this article, we introduce a TS-based robot motion-planning algorithm, which is in fact the first of its kind, as we did not find any prior instance in the literature. One reason might be that a straightforward way for defining tabu and non-tabu moves is via discretizing the C-space, for which some approaches like PFs have been introduced long ago. Nevertheless, because of our different approach in defining neighborhoods, employing TS has been possible and effective.

The Motion Planner's Components

The new online motion planner presented in this article incorporates the robot's sensory data into the intelligence induced from the TS technique.

Before dealing with the major components of the motion planner, we define a move: a move is a motion from the current point x to another point inside the free C-space (C_{free}) with a step size equal to the radius of the current point's locally maximal disc (LMD)—which is the largest disc centered around x and completely contained in C_{free}—and a direction along one of its radial sensors (Figure 1).

The outline of the algorithm is as follows. Beginning from the start point, the robot performs a visibility scan to find visible obstacle vertices and decides to move toward an obstacle vertex it finds most promising according to a cost criterion. Upon making the move, backward directions are labeled as tabu and excluded from the set of next promising directions. The visibility scan and its ensuing operations are repeated for the new location, and the robot continues to navigate the environment until it sees the goal point. If at any stage the robot is trapped in a local minimum, it takes a random and relatively large step toward unexplored areas of the search space and continues its search in that area. The algorithm's main components are described later.

Perception Component

The perception component is responsible for acquiring information from the environment and processing those data to determine the appropriate moves for the robot at each iteration. This is done by 1) performing a visibility scan and 2) detecting the visible obstacle vertices.

Upon arriving at a new point in the workspace, the robot first determines its distance to the surrounding obstacles by means of its radial range-finder sensor readings, which yields a list of candidate moves. Suppose that a circular mobile robot with radius R_{rob} and S range-sensors situated equidistantly on its perimeter is centered at point c. Each sensor projects a ray r_i ($i = 1, \ldots, S$, counterclockwise) to find out its distance ρ_i from the nearest visible obstacle point $\mathbf{x}_i$ along the i-th direction (Figure 1).

Taking the metric $D(\mathbf{x}_i, \mathbf{x}_c)$ for the Euclidean distance of points $\mathbf{x}_i$ and $\mathbf{x}_c$, we have $\rho_i = D(\mathbf{x}_c, \mathbf{x}_i) - R_{\text{rob}}$, where $\mathbf{x}_c$ is the coordinate of the robot center's current position in the workspace. A representation of ρ_i's versus ray angles is depicted in Figure 2(a).

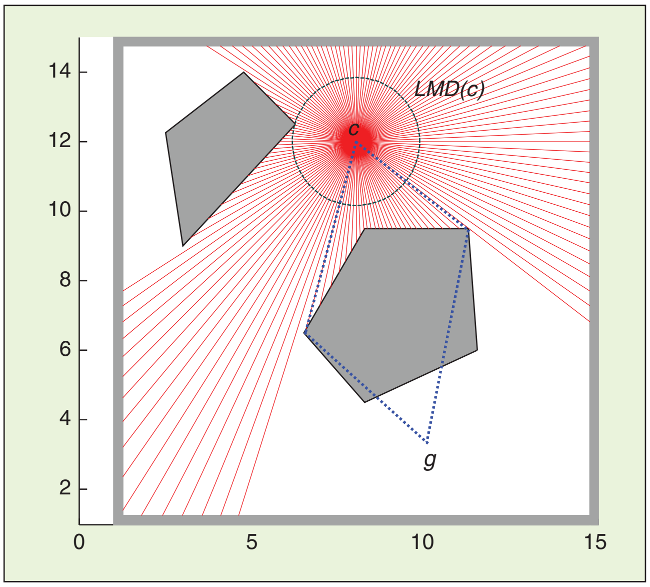

Figure 1. *The visibility scan of the environment from the robot's location at point c.*

In order for the robot to avoid getting trapped in obstacles' concave regions and bypass any blocking obstacle, it should move toward the tangent rays of the obstacle's boundaries. A ray r_i is tangent to an obstacle if in a neighborhood U of $\mathbf{x}_i$ the interior of the obstacle lies entirely on a single side of the line r_i. Otherwise, the robot's motion toward the middle of the obstacle will lead to collision. This strategy stipulates the robot to distinguish the obstacle's outermost vertices, or in a broader sense (if the obstacles are not polygons), the regions adjacent to tangent rays, as viewed from the robot's vantage point.

For determining the tangent rays, a difference function is applied for successive adjacent rays to calculate the ray difference variables, as

$$\hat{\rho}_i = \rho_{i+1} - \rho_i. \tag{1}$$

Figure 2(b) shows the difference variables of the Figure 2(a). The sharp peaks (both positive and negative) imply abrupt and large differences in successive ray magnitudes, and so indicate the points where sweeping rays leave or meet a convex contour on the obstacle boundary. These peaks are detected by applying a notch filter to the plot. If no peaks are found, then the algorithm shifts to the diversification mode in which a random step is taken by the robot.

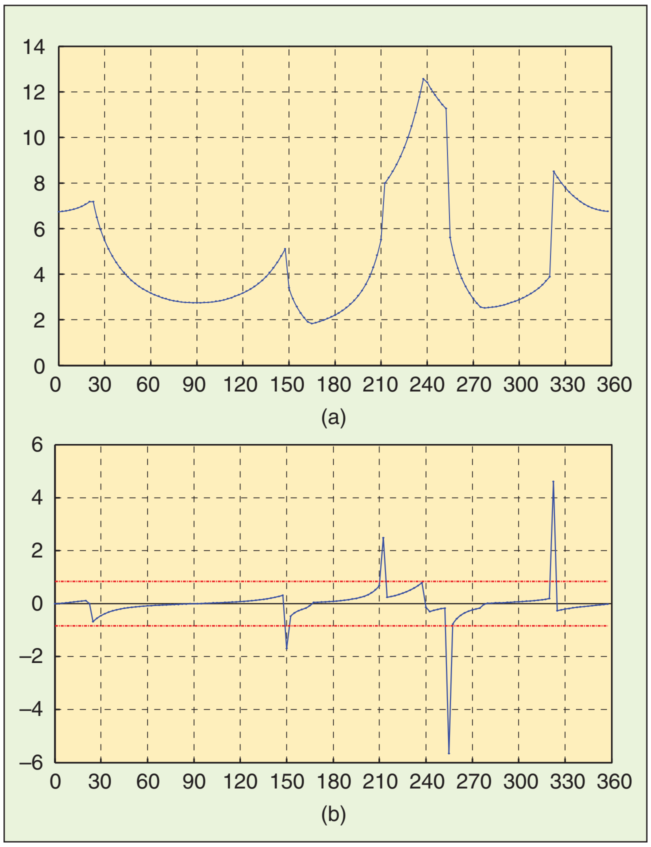

Figure 2. *(a) The magnitudes of rays emitted from the robot's position in Figure 1, which was acquired by range sensors. (b) By applying a difference function, obstacle vertices are identified by sharp peaks. Insignificant peaks are omitted by using a notch filter (dashed horizontal lines at ±0.9).*

Cost Evaluation Component

After determining visible obstacles' extreme vertices, the cost evaluation component associates a value to each tangent ray as a measure for the cost of reaching the goal via the direction of that ray. The criterion by which the cost is specified is a compound function. It incorporates two basic functions related to each ray: 1) distance function and 2) neighborhood function.

The distance function $f_D(r_i)$ aims to estimate the length of the path connecting the robot center's current configuration to the goal configuration and is defined by

$$f_D(r_i) = \lambda_1 \cdot D(\mathbf{x}_c, \mathbf{x}_i) + \lambda_2 \cdot D(\mathbf{x}_i, \mathbf{x}_g). \tag{2}$$

The first part of (2) is deterministic and is calculated in the perception component. The second term is a heuristic estimation for the length of the free path connecting the r_i's endpoint $\mathbf{x}_i$ and the goal point $\mathbf{x}_g$. The weighted linear combination of these two terms (with λ_1 and λ_2 as weights) provides a heuristic criterion widely used in the A* search technique. The thick dotted lines in Figure 1 show the distances involved in building the cost function: the lines originated from the robot's configuration show the tangent rays, i.e., the perceived obstacle vertices, and the lines to the point g present rough estimations for the distance of the vertices to the goal point, as in the A* search.

The neighborhood function $f_N(r_i)$ measures the degree of change in the magnitudes of neighboring rays occurring at r_i and is expressed as

$$f_N(r_i) = \alpha \cdot \max\{\hat{\rho}_i, \hat{\rho}_{i-1}\}, \tag{3}$$

where α is a tuning parameter. A large value of $f_N(r_i)$ implies that the obstacle (vertex) adjacent to r_i has a relatively large free space behind it and will possibly lead the robot to a key position in the configuration space, hence offering a better maneuverability for it. Small amounts of $f_N(r_i)$ indicate cramped areas, narrow passages, or obstacle borders, which generally have lesser priority for navigation.

The overall cost evaluation criterion $C(r_i)$ is minimizing a blend of the distance and neighborhood functions according to

$$C(r_i) = P_i \cdot f_D(r_i)^{\beta} \cdot f_N(r_i)^{-\gamma}, \tag{4}$$

in which β and γ are scalars and P_i is defined as

$$P_i = \begin{cases} e & \text{if } r_i \text{ has the direction of the last move} \\ v & \text{if } r_i \text{ points to a visited vertex} \\ t & \text{if } r_i \text{ is a tabu direction.} \end{cases} \tag{5}$$

Through its reducing effect, the parameter $e < 1$ encourages the robot to continue its navigation along a direction selected in the past few iterations and to be not diverted frequently by every new vertex that appears in its scope. The parameter v increases the cost of a ray pointing to a previously visited vertex, whereas the parameter t imposes a penalty for directions that are designated as tabu ones. The suggested values for these parameters are shown in Table 3.

After evaluating all rays, the motion planner is able to select the most promising goal-oriented direction to move along, which corresponds to the ray associated with the lowest cost.

Note that since the neighborhood function $f_N(r_i)$ has a negative exponent in (4), its large values (corresponding to tangent rays) reduce the overall cost dramatically. It follows that the probability of selecting obstacle vertices as next promising destinations is much higher than that of the ordinary rays (which point to obstacle borders), and, therefore, the robot is naturally being attracted to vertices. On the other hand, the distance function $f_D(r_i)$ in (4) increases the probability of selecting near-to-goal destinations through its positive exponent. In other words, the designed cost evaluation function leads to locally optimal (i.e., shortest) navigations, just as the visibility graph does in offline mode, and, thus, the robot's performance improves significantly.

Aspiration and Desperation Levels

The aspiration level is a level set to accept a very good move, even if it is tabu. This is an established concept in TS, proposed by Glover. Now, we make the TS metaheuristic more flexible and powerful by introducing a new concept called desperation level.

The desperation level is a level (of cost) beyond which a nontabu move having higher cost values (for minimization problems) or lower cost values (for maximization problems) is rejected and included in the tabu list. It is somehow a counterpart and complementary concept for the aspiration level. The relation of these levels with regard to the tabu or nontabu moves is depicted in Figure 3. The gray sections of the diagram (i.e., II and IV) represent unacceptable moves, since they are either tabu—with costs not better than the aspiration level—or nontabu, but with costs higher than the desperation level.

In our planning context, it frequently happens that there are no nearby nontabu obstacle vertices. Instead, there are some remote vertices with high costs beyond the desperation level. Excluding such vertices from the moves list will make the list empty and limit the search space, which in turn will activate the diversification component and will cause the robot to take a random step toward unexplored areas of the space.

The Short-Term Tabu List

The notion of a tabu list is critical and fundamental to this approach. The attribute by which we set up tabu lists is the direction of the rays emanated from the robot.

In each iteration, tabu moves are identified based on the robot's direction. A tabu envelope (TE) variable is specified to set the range of tabu directions. It is laid out symmetrically around the reverse of the robot's direction, covering all rays within a $\pm$TE/2 deviation. For instance, in Figure 4, the robot's initial direction is 50°. By setting TE = 90°, all the directions included in the area $(\pi + 50°) \pm 45°$ (i.e., 185–275°) are characterized as tabu moves (the gray sector).

For setting up the short-term tabu list, if its size (STLS) is set to k, then the tabu directions of the last k iterations are appended to form the total set of tabu moves. We found $k = 2$ to be the best size, although $k = 1$ is also possible, but it leads to more fluctuations in the robot's motion.

The Long-Term Tabu List

Aside from the short-term tabu list discussed earlier, we set a long-term tabu list by keeping a record of the already visited (i.e., almost touched) vertices. If the robot happens to head for a visited vertex, then since it has been at that location during earlier iterations, it should avoid the point and concentrate on other vertices in view. This is done by a simple checking of the long-term tabu list, with a size of LTLS. The long-term tabu list may also contain a set of nonvertex points that are proven to be ineffective and misleading in directing the robot toward the goal.

Diversification Component

This component is evoked when there are no admissible nontabu directions. This situation occurs when 1) all the vertices in scope have been previously visited and marked as tabu (i.e., are in tabu list), 2) all nontabu moves have a cost value higher than the desperation level, or 3) the robot is entrapped in a dead end.

The robot will then take a large step with a random direction selected from among rays with big magnitudes (compare with the concept of the diversification phase discussed earlier). The short-term tabu list is cleared after this step, but the long-term tabu list is retained. This action will most likely guide the

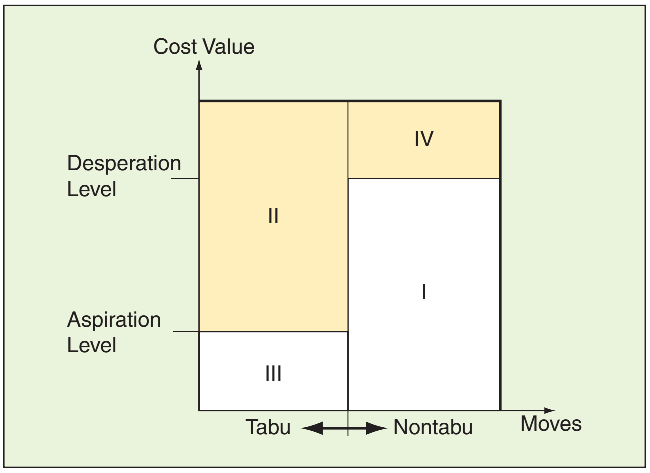

Figure 3. *The aspiration and desperation levels for a minimization problem. Region IV represents moves considered unacceptable, although nontabu.*

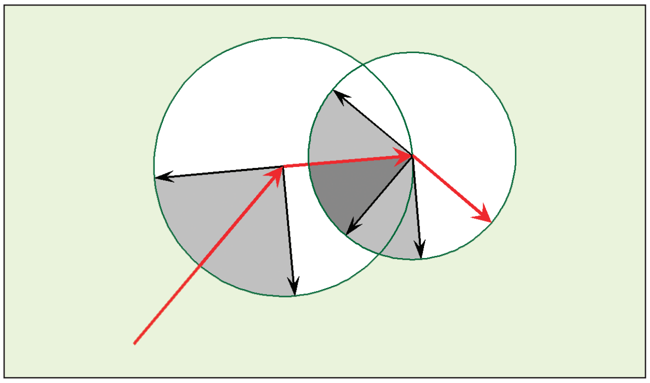

Figure 4. *A short list of tabu moves is constructed by appending the tabu envelopes of the last two iterations.*

robot toward new and unexplored areas of the searching space. Consequently, all the elements of the short-term tabu list, as well as some older elements of the long-term tabu list, will be eliminated.

In fact, the diversification component provides the planner's probabilistic-completeness property. That is, given sufficient time, the algorithm will eventually reach the goal if there is a valid path. The random steps guarantee that the robot will explore all areas of the workspace.

Safety Component

When the robot approaches an obstacle border closer than a preset safety radius R_s, it should take a reflective step away from the obstacle to maintain its safety. This is similar to the behavior of a light beam when reflected from a surface. The reflective step in Figure 5(a) directs the robot toward a safer location via a relatively large and outward movement. It also effectively helps the robot to turn around obstacle vertices and sharp corners [Figure 5(b)].

The length of the reflective step is set to a few times the radius of the LMD, and its direction is determined by a summation of the robot's direction vector and the obstacle border's normal vector.

Goal Connection Component

If the goal point lies within the sights of the robot, then through a goal connecting operation, the robot's location is connected to the goal point via a straight line. The robot then has to follow that line and terminate its search.

Algorithm Steps

By integrating the aforementioned components in a single architecture, the TS-based online motion planner follows these steps to produce a goal-driven trajectory:

Step 1: The goal connection component checks whether the goal is visible: if it is visible, the current point is connected directly to the goal, and the algorithm is terminated. Otherwise, go to Step 2.

Step 2: The robot activates the perception component, including the visibility scan and the discovery of surrounding obstacles' vertices.

Step 3: The cost evaluation component evaluates the cost of all directions based on a criterion.

Step 4: Update the short-term tabu list (based on the tabu envelopes of the last two moves) as well as the long-term tabu list (based on visited obstacle vertices or misleading configurations).

Step 5: If an obstacle border is closer than the safety radius, the robot takes a reflective step away from the obstacle border and goes to Step 1. Otherwise, go to Step 6.

Step 6: Construct a list of nontabu directions by excluding tabu moves from the set of all moves and considering the aspiration and desperation levels. If the nontabu list is not empty, select a direction with the lowest cost among the nontabu moves. Otherwise, activate the diversification component and take a large step along a random direction. Go to Step 1.

Experimentation

The algorithm was run for several problems ranging from simple convex to highly concave polygons and mazes and succeeded in performing effectively. Some of the simulations are shown in Figure 6(a)–(h). The running times were within a few seconds using a 2.16 GHz Intel Duo processor.

The robot navigates faster in sparse and uncluttered areas and more cautiously in cluttered and near-to-obstacle regions. The sharp angles in the trajectories are due to the reflective steps. The effect of the diversification component can be seen in Figure 6(h), where the upper-left large step is the random move made after backtracking, hoping for exploring new areas.

To test the efficiency of the proposed method and compare it with other approaches, we designed and solved a number of test problems. The results are shown in Table 1. Compared with global optimum solutions, the paths produced by the TS-based planner had 9.25% average error. Large errors generally occurred in maze-like problems. Because of the vertex attraction fact imposed by the adopted cost function, the planner has a tendency to follow short paths inherited from the visibility graph roadmap (which produces the shortest path in offline mode). Therefore, as an online method, the path quality is quite satisfactory, especially when compared with other offline methods like A* grid search, PF, or Voronoi diagrams methods (Table 1 and Figure 7). Because grid-based methods (like PF and A*) examine the neighboring cells of grid points, the resulting path is rough and can only have vertical, horizontal, and diagonal local directions.

Since online methods acquire their knowledge of environment by sensors and plan their path locally, it would be incorrect to compare the processing times of offline and online methods. However, compared with the average

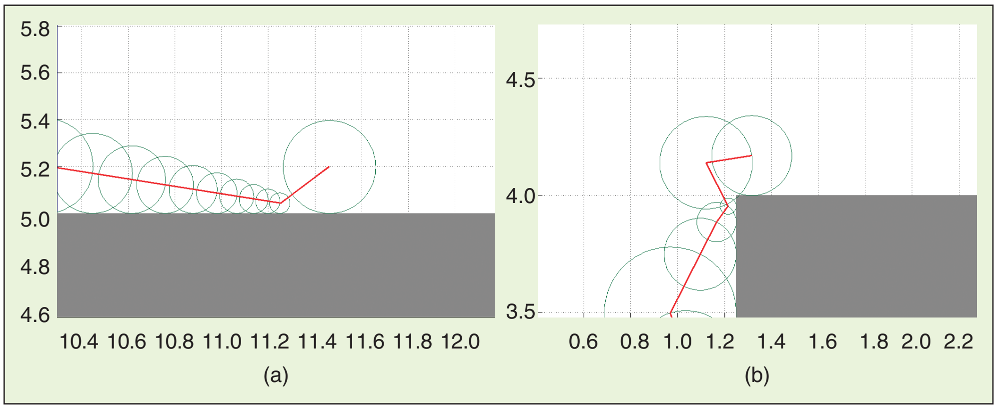

Figure 5. *(a) The reflective step guarantees that the robot will keep a safe distance from the obstacles. (b) The reflective step enables vertex surmounting.*

processing time of 7.46 s for the A★ search and 3.08 s for the PF method (calculated for the 15 problems), the TS-based algorithm performed convincingly well.

We also compared the performance of the proposed new planner with that of an online distance transform method, the incremental construction of generalized Voronoi diagram (GVD). This method builds the GVD incrementally using the information acquired by its sensors [6], [7]. Overall, the path lengths of the TS-based method are shorter (about 25% less in our experiments) than that of the GVD method. This is due to the nature of the Voronoi diagram that keeps the maximum clearance from the obstacles, whereas the TS-based planner is attracted to obstacle vertices and thus emulates the visibility graph, which provides the shortest path. Problem 13 in Table 1 was solved by the GVD online method in 24.3 s through 243 iterations and with 24 sensors and a path length of 75.06

The robot navigates faster in sparse and uncluttered areas and more cautiously in cluttered and near-to-obstacle regions.

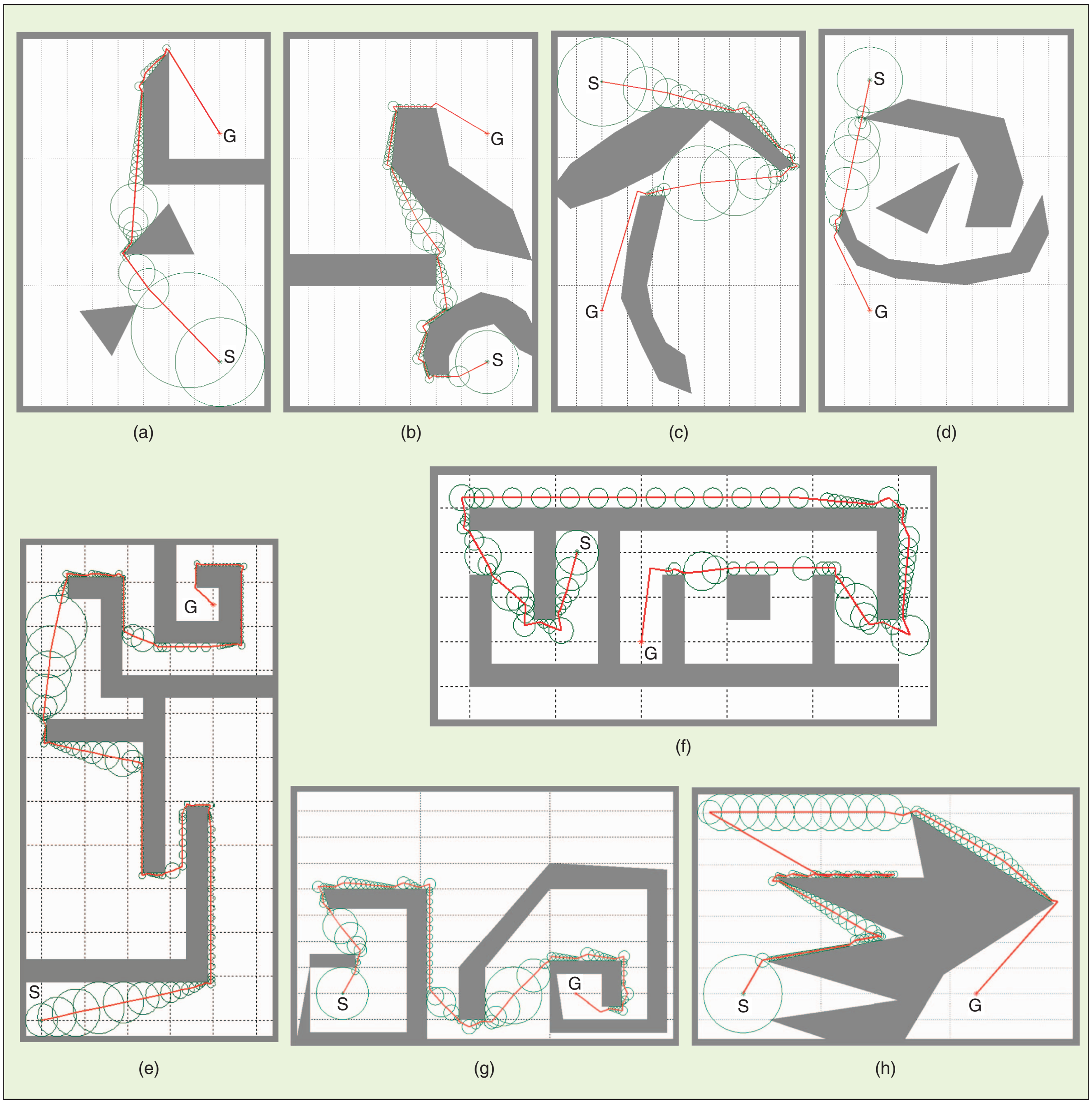

Figure 6. *Some experiments: (a)–(d) convex and concave obstacles, (e)–(g) maze-like obstacles, (h) a random move is made following the activation of the diversification component.*

[Figure 7(c)]. The TS-based algorithm's solution is shown in Table 1 and Figure 6(e). Other problems were also compared and gave more or less the same results.

To evaluate the performance of the TS-based planner against one more online motion planner, we selected the sensor-based rapidly exploring random tree (SRT) method [15], which is an online version of LaValle's rapidly exploring random tree (RRT) method [16]. The SRT-Star method was run for the 15 benchmark problems, and the results are summarized in Table 2.

As its name implies, the SRT method builds a rooted tree from the start point, and at each iteration, by generating neighboring nodes, it extends its branches randomly but toward previously unexplored areas of the C-space. When encountering

Table 1. A comparison of the path lengths generated by different methods.

			TS-Based Online Planner				Path Lengths by Offline Methods				TS-Based
Problem	Number of Convex Obstacles	Work-Space Size	Number of Sensors	Number of Iterations	CPU Time (s)	Path Length	A* Search[ab]	Potential Fields[ac]	Voronoi Diagram[d]	Visibility Graph[e]	Path Length Error %
1	1	[10 × 10]	24	7	0.24	10.42	10.61	13.31	14.07	10.33	0.87
2	2	[10 × 10]	24	31	1.37	14.08	14.18	21.74	19.90	13.72	2.62
3	4	[10 × 10]	36	60	4.11	18.85	21.53	24.67	27.71	18.01	4.66
4	5	[15 × 10]	16	36	1.65	14.94	18.27	20.85	21.92	13.92	7.33
5	5	[15 × 10]	16	47	4.09	21.49	22.26	28.77	33.01	19.12	12.39
6	6	[10 × 10]	18	79	7.00	22.64	23.21	27.22	26.82	18.96	19.41
7	7	[15 × 10]	24	40	2.54	17.77	19.53	23.79	26.66	16.95	4.84
8	8	[10 × 10]	24	41	4.20	18.09	18.06	18.81	20.37	15.38	17.62
9	8	[15 × 10]	36	46	5.31	15.74	17.75	21.95	20.47	15.18	3.69
10	7	[10 × 10]	36	83	9.79	25.54	26.25	34.51	31.84	23.73	7.63
11	11	[15 × 10]	20	53	7.36	19.24	18.48	21.94	24.35	16.36	17.60
12	16	[15 × 10]	36	29	10.10	14.07	13.55	18.58	17.40	13.93	1.01
13	11	[13 × 24]	20	183	14.06	64.67	54.50	64.90	74.61	53.60	20.65
14	12	[9 × 10]	24	52	4.62	13.04	19.11	18.46	16.31	12.26	6.36
15	16	[13 × 24]	40	97	23.86	32.43	31.71	36.31	39.46	28.90	12.20
Average				59	6.68	21.53	21.93	26.39	27.66	19.36	9.25

[a]After graduating the workspace with 1/10 of the unit length.
[b]With a best-first search strategy and Euclidean distance heuristic.
[c]With filling-up local minima.
[d]Searched by the Dijkstra's method.
[e]Optimal solution.

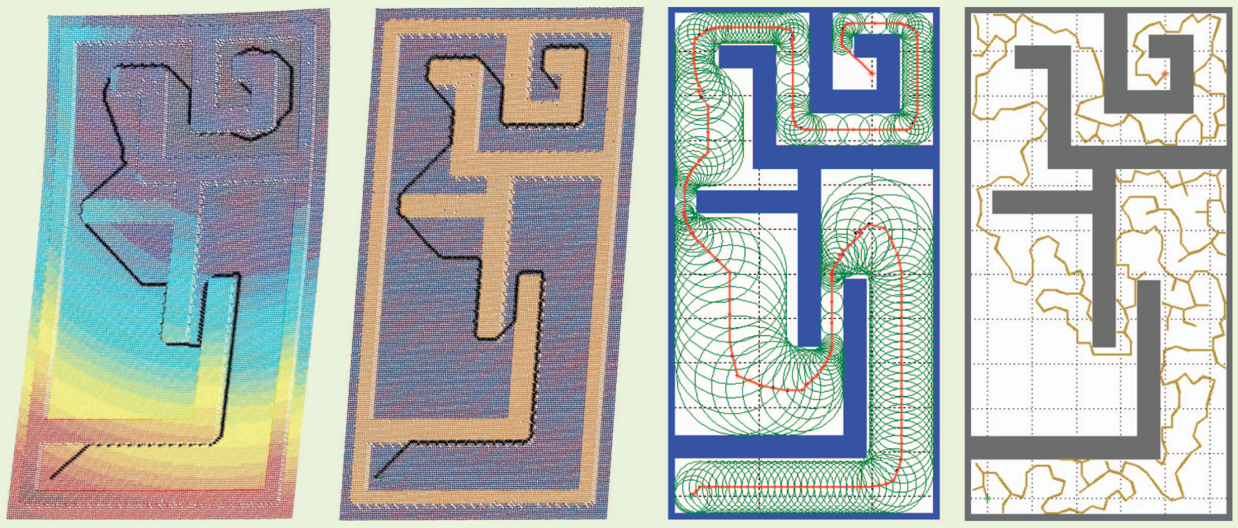

Figure 7. *The problem 13 in Table 1 is solved by (a) PF approach in 3.1 s, (b) A* search in 41.5 s, (c) online distance transform (GVD) builder in 24.3 s, and (d) sensor-based RRT planner in 14.4 s.*

a dead end, it backtracks to its parent node and repeats the procedure, until filling the whole C-space. SRT is essentially developed for workspace exploration, and not for goal reaching. So, we modified it slightly to terminate the search if the goal point is viewed by sensors. A sample output for the problem 13 is illustrated in Figure 7(d).

We observed that SRT fails to reach the goal at some runs, despite calibrating its parameters carefully. This is because it is a resolution- and probabilistic-complete method. Since the time and path length vary greatly because of the SRT's highly stochastic nature, we did several experiments for providing sufficient valid data for our comparison. The failure rate (%), average number of iterations, as well as the mean and standard deviation of the results are also included in Table 2. The last two columns show relative time and path lengths of the TS-based and SRT planners. In both aspects, values less than 1 show the TS-based planner's fine performance.

Parameter Setting and Tuning

As a metaheuristic approach, the TS technique requires proper setting of its parameters.

A key parameter is the number of sensors: it is assumed that the robot has a circular perimeter equipped with a sensor ring. Practically, the number of sensors is less than 30 or 40. However, for our theoretical investigations and simulations, we selected different sensor numbers such as 12, 18, 24, 36, 40, 60, 72, 90, 120, 180, 240, and 360. The larger is the number of sensors, the more exact is the perception of the environment. On the other hand, large sensor numbers require high computational time and memory.

In the case that the mobile robot (or its sensor) has the ability to rotate about a central axis, then the number of radial sensors becomes less critical, since a disc-shaped robot, for instance, can multiply its environmental perception by i times

The tabu search method, which is a well-known metaheuristic technique for solving difficult combinatorial optimization problems, is being applied in robot motion planning for the first time.

$(i = 2, 3, \ldots)$ via rotating about its center with increments of $2\pi/(S \times i)$ degrees. A sensitivity analysis for the number of sensors can help users determine the proper degrees of rotation, as well as the hardware requirements for the robot's successful motion planning and navigation. The SRT method proved to be less sensitive to the number of sensors.

A summary of the parameters used in the algorithm, together with our suggested values for them, is presented in Table 3. These values are set through extensive tests and evaluations.

The exploration procedure would be successful if these parameters are set properly suitable for the workspace under navigation. To reduce the risk of improper parameter setting for unknown environments, and to make the planner even more intelligent, we propose a number of rules and guidelines associated with each parameter, such that as the exploration goes on, the robot can adapt itself to different situations by learning more and more about the environment. Thus, it can adopt strategies for reaching the goal, avoiding local minima, and tuning its parameters automatically. We have discussed the effects of tuning each parameter in Table 3. This automatic adaptation makes the TS-based planner very efficient and powerful.

Table 2. Results of solving the test problems with SRT.

Problem	Average Number of Iterations	Average Number of Nodes	CPU Time		Path Length		Failure Rate (%)	TS Time Versus SRT Time	TS Path Versus SRT Path
			Average	Standard Deviation	Average	Standard Deviation			
1	28.6	23.7	0.94	0.38	32.92	9.74	0.0	0.25	0.32
2	34.9	27.3	1.20	0.23	37.72	5.63	8.3	1.15	0.37
3	67.8	58.5	3.43	0.53	55.74	6.67	9.1	1.20	0.34
4	119.8	84.8	5.46	2.73	81.60	23.18	0.0	0.27	0.18
5	127.4	94.7	7.21	2.32	90.48	19.18	23.1	0.57	0.24
6	78.5	65.0	3.97	0.84	61.94	8.10	9.1	1.76	0.37
7	147.2	98.8	8.84	3.28	93.84	22.16	9.1	0.29	0.19
8	81.9	58.1	4.49	1.04	54.35	9.02	37.5	0.94	0.33
9	121.8	86.8	10.02	5.48	82.74	24.22	8.3	0.53	0.19
10	71.3	60.6	5.22	0.99	56.52	6.94	50.0	1.88	0.45
11	109.0	77.2	8.48	2.79	72.41	16.87	16.6	0.87	0.27
12	105.4	77.9	18.74	7.89	74.73	23.82	0.0	0.54	0.19
13	254.6	203.7	18.39	8.37	183.60	26.21	65.5	0.76	0.35
14	57.9	45.2	3.28	1.51	41.36	15.16	36.8	1.41	0.32
15	249.7	177.3	38.86	17.97	170.60	45.92	7.1	0.61	0.19
Total average			9.24	3.76	79.37	17.52	18.7	0.72	0.27

The aspiration level is a level of cost set to accept very good moves even if they are tabu, whereas the desperation level is a level of cost beyond which nontabu moves are rejected.

Discussion

Two very important issues of a path planning algorithm are its time complexity and completeness.

For determining the time complexity of the algorithm, we should first estimate the number of iterations required to accomplish the path planning task by establishing an upper bound for it. The worst condition occurs when the robot takes smallest possible steps all the time. This happens if the robot moves along the obstacle borders with a minimum clearance, determined by the value of the safety radius, while visiting all the obstacles in the workspace. Suppose that there are m disjoint obstacles that are arranged in a regular array. Taking the overall number of obstacle vertices as n, the total number of obstacle edges would also be n, and the total border length would be finite times n, plus the distances of interobstacle traversals, which is finite times m. Since $m < n$, the upper bound of the maximum number of iterations is in $O(n)$. Even if this length is navigated with the smallest possible step size (R_s), because of the constant number of computations in each iteration, the time complexity of the algorithm would still be in $O(n)$.

Unlike distance transform planners (which explore the medial axis of the C-space thoroughly), the TS-based planner does not benefit from a similar connected graph, and so it is not complete. However, because of the large diversifying steps taken on a random basis, the robot can explore all unexplored areas given sufficient time, and so is probabilistically complete, which means it is guaranteed to reach the goal within a long time.

Conclusions

The new online motion planner developed in this article is based on the tabu search metaheuristic. Various components

Table 3. Parameters used in the motion planner.

Symbol	Description	Suggested Range	Conditions for an Increase	Conditions for a Decrease
TE	Tabu envelope	$[\pi/2, \pi]$	The workspace is uncluttered and straight motions would work well	There are many narrow passageways
STLS	Short tabu list size	1 or 2	Must not exceed 2	Cannot find enough vertices
LTLS	Long tabu list size	[5, 20], integer	The number of vertices is large, or the goal is far from current location	Otherwise
AL	Aspiration level	[0.2, 0.8]	Frequent turn-backs are required	A look-forward approach is selected
DL	Desperation level	[5, 15]	Exploring a semiclosed area	Escaping an area by random moves
λ_1, λ_2	Distance function coefficients	(0, 1]	$\lambda_1/\lambda_2 > 1$: more predictive than realistic, for conventional areas	$\lambda_1/\lambda_2 < 1$: more realistic than predictive, for unpredictable areas
α	Neighborhood function coefficient	(0, 1]	Exploring vast free areas located behind obstacles	Focusing on reaching the goal via narrow openings and passages
β	Total cost function coefficient	(0, 3]	Moving away from the goal before approaching it (e.g., when getting out of a cul-de-sac)	A more goal-oriented, "greedy" action is required
γ	Total cost function coefficient	(0, 3]	Exploring vast free areas located behind obstacles	Focusing on reaching the goal via narrow openings and passages
e	Straight move incentive	[0.35, 1]	Reacting to (and approach) new obstacles	The trajectory fluctuates much and is not smooth
v	Vertex revisiting penalty	[6, 10]	Cannot find the goal and want to take more random steps	Reexploring a previsited area
t	Tabu direction penalty	[4, 8]	Exploring unvisited areas	Intensifying exploring an area
R_S	Safety radius	$[1.1, 1.5] \times R_{rob}$	More safety is required	Otherwise
R_f	Reflective step length	$[4, 8] \times R_S$	Taking more risk of collision	A wall-following behavior is required

of the classic TS have been remodeled and integrated in a single algorithm to craft a motion planner capable of solving varieties of exploration and goal-finding problems. By employing different combinations of a number of parameters, the planner can react intelligently and promptly to the new situations it faces during the robotic navigation. The presented explanations on the parameters' definitions and attributes can help researchers in applying this algorithm to their real-world experiments and applications.

The newly defined concept of desperation level also enriches the still-evolving TS discipline, and together with the aspiration level and the diversification step, it enables the robot particularly to escape from local minima. Numerous experiments and comparisons with offline and online methods showed the algorithm's success and efficiency in coping with different problems, from simple polygons to highly concave obstacles.

Considering the online and sensor-based nature of the presented model, it is believed that it can be applied to dynamic environments (with moving obstacles) as well. In that case, the neighborhood and distance functions must be modified to accommodate some predictive information about the velocity vectors of each obstacle.

Considering the online and sensor-based nature of the presented model, it can be applied to dynamic environments as well.

Keywords

Robot motion planning, sensor-based navigation, tabu search metaheuristic.

References

[1] V. J. Lumelsky and A. A. Stepanov, "Dynamic path planning for a mobile automation with limited information on the environment," *IEEE Trans. Automat. Contr.*, vol. 31, no. 11, pp. 1058–1063, 1986.
[2] J. Cox and C. K. Yap, "On-line motion planning: moving a planar arm by probing an unknown environment," Courant Institute of Mathematical Sciences, New York Univ., New York, Tech. Rep., July 1988.
[3] N. S. V. Rao, S. Kareti, W. Shi, and S. S. Iyenagar, "Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms," Oakridge National Lab., Tech. Rep. ORNL/TM-12410, 1993.
[4] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. Robot. Res.*, vol. 5, no. 1, pp. 90–98, 1986.
[5] N. S. V. Rao, N. Stolzfus, and S. S. Iyengar, "A 'retraction' method for learned navigation in unknown terrains for a circular robot," *IEEE Trans. Robot. Automat.*, vol. 7, no. 5, pp. 699–707, 1991.
[6] H. Choset, S. Walker, K. Eiamsa-Ard, and J. Burdick, "Sensor-based exploration: Incremental construction of the hierarchical generalized Voronoi graph," *Int. J. Robot. Res.*, vol. 19, no. 2, pp. 126–148, 2000.
[7] E. Masehian, M. R. Amin-Naseri, and S. Esmaeilzadeh Khadem, "Online motion planning using incremental construction of medial axis," in *Proc. IEEE Int. Conf. Robotics and Automation*, Sept. 2003, vol. 3, pp. 2928–2933.
[8] V. Krishnaswamy and W. S. Newman, "Online motion planning using critical point graphs in two-dimensional configuration space," in *Proc. IEEE Int. Conf. Robotics and Automation*, May 1992, vol. 3, pp. 2334–2339.
[9] J. B. Oommen, S. S. Iyengar, N. S. V. Rao, and R. L. Kashyap, "Robot navigation in unknown terrains using visibility graphs—Part I: The disjoint convex obstacle case," *IEEE J. Robot. Automat.*, vol. 3, pp. 672–681, Dec. 1987.
[10] J. M. Ahuactzin, T. El-Ghazali, P. Bessiere, and E. Mazer, "Using genetic algorithms for robot motion planning," in *Proc. Workshop of Geometric Reasoning for Perception and Action*, Sept. 16–17, 1991, pp. 84–93.
[11] C. Eldershaw and S. Cameron, "Real-world applications: Motion planning using GAs," in *Proc. Genetic & Evolutionary Computation Conf.*, Orlando, FL, July 1999, p. 1776.
[12] F. Janabi-Sharifi and D. Vinke, "Integration of the artificial potential field approach with simulated annealing for robot path planning," in *Proc. IEEE Int. Symp. Intell. Contr.*, Aug. 1993, pp. 536–541.
[13] F. Glover, "Tabu search—Part I," *ORSA J. Comp.*, vol. 1, no. 3, pp. 190–206, 1989.
[14] F. Glover and M. Laguna, *Tabu Search*. Norwell, MA: Kluwer Academic, 1997.
[15] G. Oriolo, M. Vendittelli, L. Freda, and G. Troso, "The SRT method: Randomized strategies for exploration," in *Proc. IEEE Int. Conf. Robotics and Automation*, New Orleans, LA, Apr. 2004, pp. 4688–4694.
[16] S. M. LaValle, "Rapidly-exploring random trees; a new tool for path planning," Comp. Sci. Dept., Iowa State Univ., Tech. Rep. TR 98-11, Oct. 1998.

Ellips Masehian is an assistant professor at the Faculty of Engineering, Tarbiat Modares University, Tehran, Iran. He received the B.S. and M.S. degrees in industrial engineering, both from Iran University of Science and Technology, Tehran, with honors, and a Ph.D. degree from Tarbiat Modares University. His research is focused on the application of heuristic and intelligent methods to single and multiple robot motion planning problems.

Mohammad Reza Amin-Naseri is currently an associate professor of the industrial engineering department at Tarbiat Modares University, Tehran, Iran. He received his B.S. degree in chemical and petrochemical engineering from Amir-Kabir University of Technology (Tehran Polytechnic), an M.S. degree in operations research from Western Michigan University, and a Ph.D. degree in industrial and systems engineering from West Virginia University. His research interests include computational intelligence, artificial neural networks, metaheuristics, and their applications to various optimization problems.

Address for Correspondence: Ellips Masehian, Faculty of Engineering, Tarbiat Modares University, Jalale-ale-Ahmad Highway, Tehran, 14115-317, Iran. E-mail: masehian@modares.ac.ir.